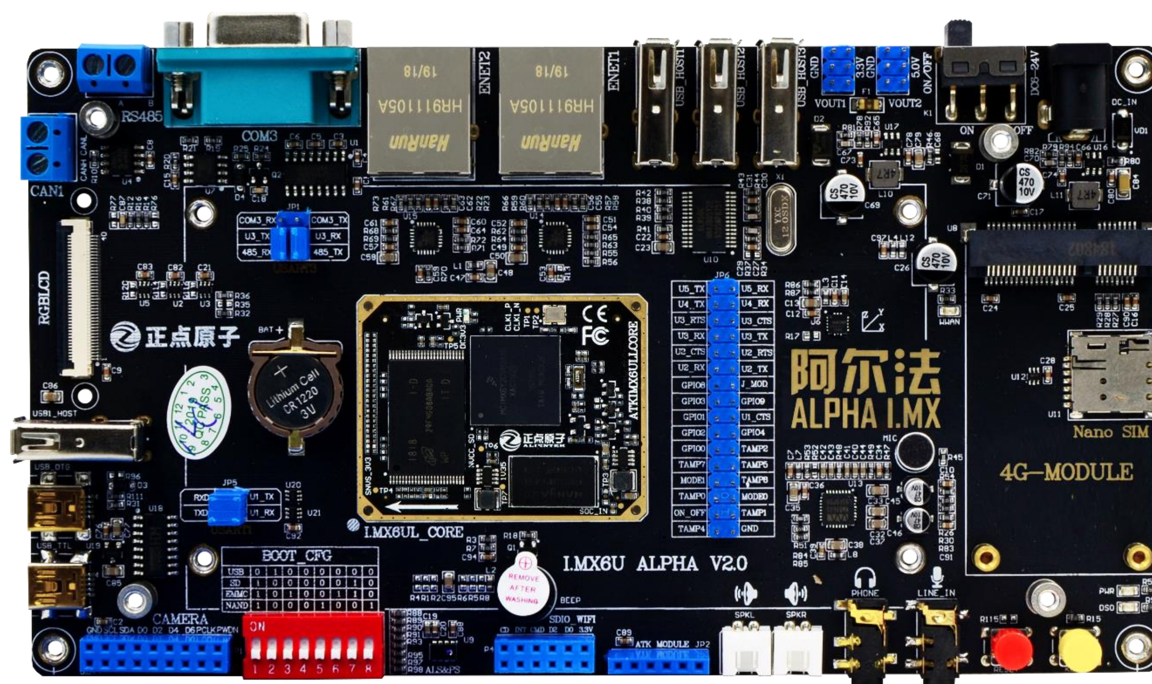
I.MX6U 用户快速体验 V1.1

-I.MX6U-ALPHA 快速体验



正点原子广州市星翼电子科技有限公司

淘宝店铺 1：http://eboard.taobao.com

淘宝店铺 2：http://openedv.taobao.com

技术支持论坛 (开源电子网) ：www.openedv.com

原子哥在线教学：www.yuanzige.com

官方网站：www.alientek.com

最新资料下载链接：http://www.openedv.com/posts/list/13912.htm

E-mail: 389063473@qq.com　　QQ: 389063473

咨询电话：020-38271790

传真号码：020-36773971

团队：正点原子团队

正点原子，做最全面、最优秀的嵌入式开发平台软硬件供应商。

友 情 提 示

如果您想及时免费获取“正点原子”最新资料，敬请关注正点原子微信公众平台，我们将及时给您发布最新消息和重要资料。



关注方法：

(1)微信“扫一扫”，扫描右侧二维码，添加关注

(2)微信→添加朋友→公众号→输入“正点原子”→关注

(3)微信→添加朋友→输入“alientek_stm32”→关注



文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 linux 团队		2019.10.26
V1.1	1. 更新 2.2.1.1 小节，修改文档 vbs 脚本名称与实际要操作的 vbs 脚本名称不对应。 2. 更新 2.2 小节，修改拔码截图和 otg 连接截图。 3. 更新 2.3 小节，修改为免输 root 登录。 4. 其他小修改。	正点原子 linux 团队		2019.11.2

目录

前言

写这个用户体验是为了帮助用户能够对板子开发板快速上手。因为是用户体验文档，介绍篇幅不长，若本文档有错漏的地方还请谅解，恳请到正点原子技术论坛指正，日后会更正和完善文档。

第一章 ALIENTEK I.MX6U 软硬件资源简介

1.1 硬件资源简介

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.1~5.2.1 小节。

1.2 软件资源简介

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.2.2 小节。

第二章 ALIENTEK I.MX6U 使用前准备

2.1 安装驱动和串口调试终端软件

说明：用户要操控开发板，需要安装串口调试终端软件，在串口调试终端里可以输入指令来调试开发板。安装串口调试终端软件前需要安装 CH340 驱动，下面介绍它们的安装方法。

2.1.1 安装 CH340 驱动

进入 3、软件->CH340 驱动(USB 串口驱动)_XP_WIN7 共用路径下，双击 SETUP.EXE 安装 USB 串口驱动。

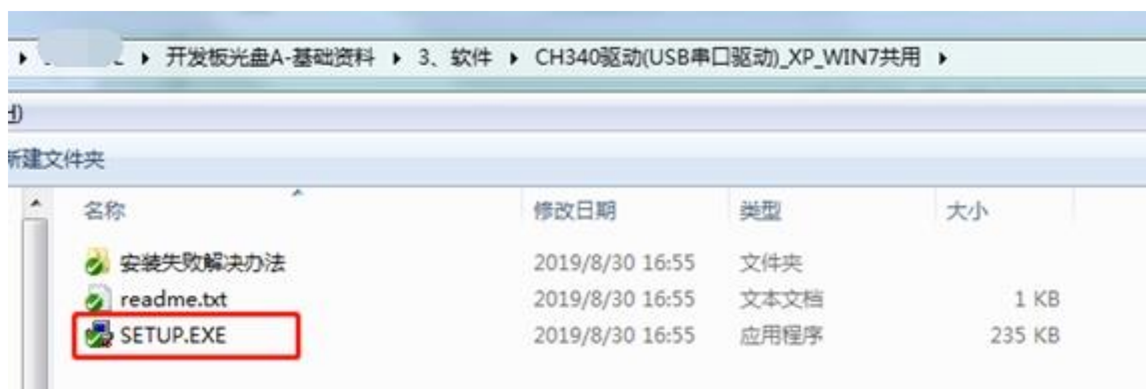


图 2.1.1.1 双击安装 CH340 驱动

点击下图 2.1.1. 2 的安装按钮，即可完成安装驱动。



图 2.1. 2 点击安装驱动

安装成功图图 2.1.1. 3 所示。



图 2.1.3 安装成功示意图

2.1.2 如何使用串口调试终端软件

进入 3、软件->SecureCRT7.1 目录下，请根据个人电脑的位数选择对应位数的串口调试软件打开。比如作者的电脑是 64 位的，就选择 scrt733-x64.exe 来安装。32 位电脑请选择 scrt712-x86.exe 来安装。安装教程请参照我们的教程《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.0.pdf》第 4.7 小节来安装使用。

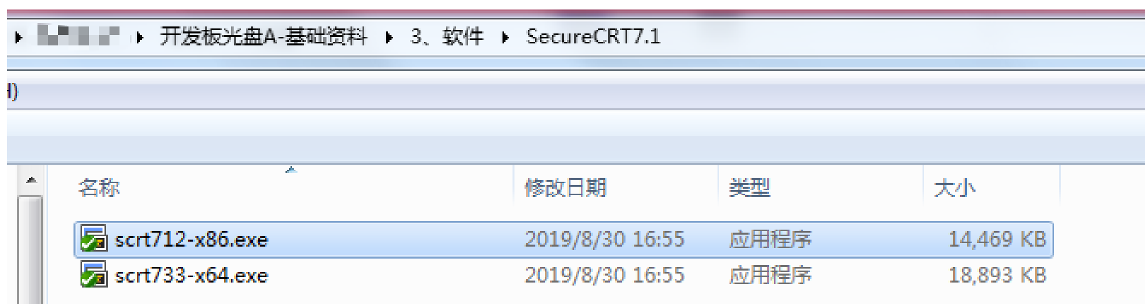


图 2.1.2. 1 双击启动串口调试软件

安装 SecureCRT7.1 软件后，开发板与 PC 机（电脑）连接，请将 USB 串口线插到开发板 USB_TTL 接口处，另一端连接电脑 USB 接口。

在设备管理器里查看 USB 串口的端口号，例如下图作者的端口号为 COM10。

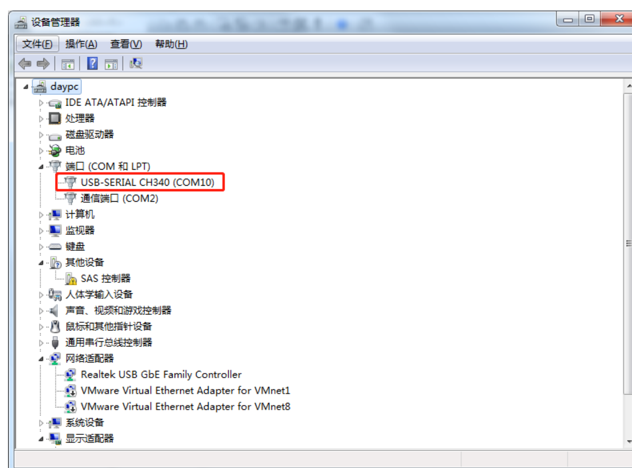


图 2.1.2. 2 查看 USB 串口的端口号

选择正确的 COM 口，波特率为 115200，8N1，无检验位，并建立串口连接。如下图图

2.1.2. 3 所示。

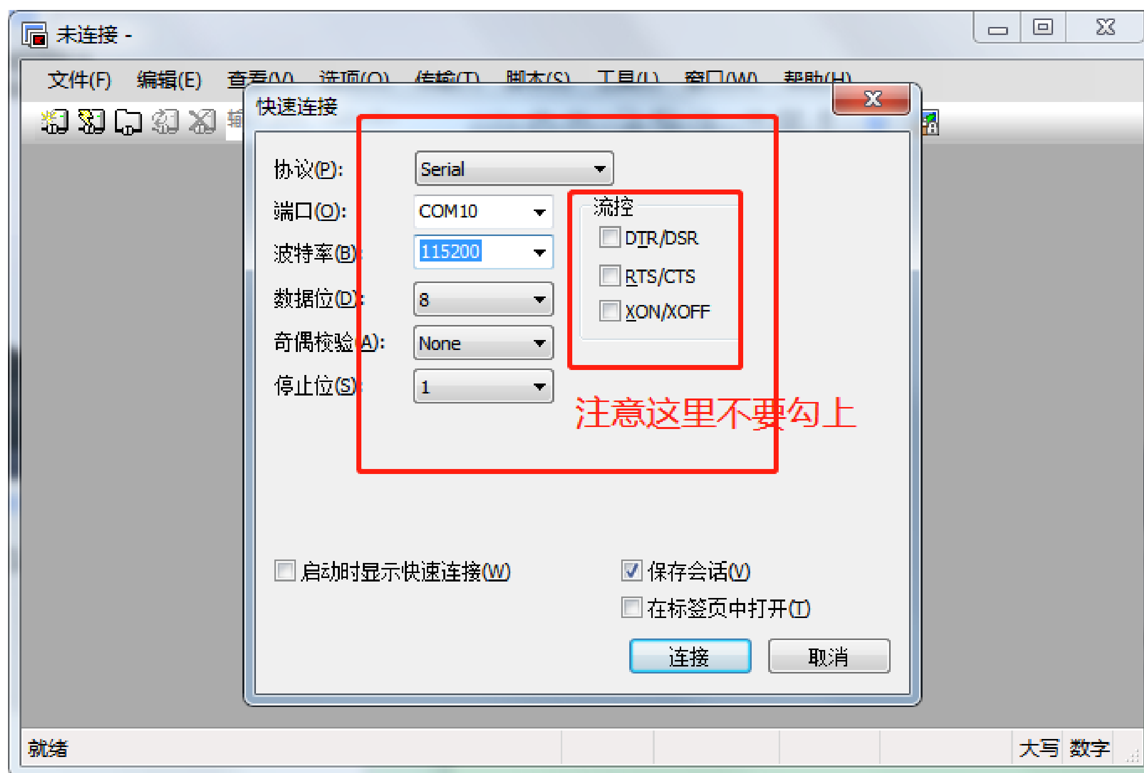


图 2.1.2. 3 串口调试软件的配置信息

设置完成后，选择对应的启动方式启动开发板就可以看到调试信息了。如不清楚启动方式的用户请看 2.3 小节。

```
serial-com10 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 窗口(W) 帮助(H)
serial-com10

U-Boot 2016.03-gfe2691b (Sep 03 2019 - 11:03:12 +0800)

CPU:   Freescale i.MX6ULL rev1.1 69 MHz (running at 396 MHz)
CPU:   Industrial temperature grade (-40C to 105C) at 45C
Reset cause: POR
Board: MX6ULL 14x14 EVK
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment

Display: ATK-LCD-4.3-800x480 (800x480)
Video: 800x480x24
reading alientek.bmp
** Unable to read file alientek.bmp **
Error: no valid bmp image at 88000000
In:    serial
Out:   serial
Err:   serial
switch to partitions #0, OK
mmc1(part 0) is current device
Net:   FEC1
Error: FEC1 address not set.

Normal Boot
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1(part 0) is current device
switch to partitions #0, OK
mmc1(part 0) is current device
reading boot.scr
** Unable to read file boot.scr **
reading zImage
6774160 bytes read in 225 ms (28.7 MiB/s)
Booting from mmc ...
reading imx6ull-14x14-emmc-4.3-800x480-c.dtb

就绪   Serial: COM10, 115200  38,  1  38 行, 80 列  VT100   大写 数字
```

图 2.1.2.4 开发板系统启动时所打印的信息

2.2 固化系统

开发板出厂已经固化系统到 eMMC/NandFlash 存储介质里。建议自行更新系统固件，正点原子会继续更新 uboot 和内核以及文件系统，修复可能存在的 bug、优化程序及添加新的功能。如果不需要重新固化系统，请直接到 2.3 节登录开发板。

正点原子提供两种固化系统方式，第一种方式是使用正点原子修改过的 NXP 官方的上位机工具 mfgtool。这种固化系统方式可以使用 PC 机在线直接固化系统。第二种方法需要制作 SD 卡系统卡，插卡的方式固化系统。这两种方法都有各自的好处。下面将介绍它们的用法。

2.2.1 使用 mfgtool 上位机固化系统（OTG 方式）

把->开发板光盘->5、开发工具->4、正点原子修改过的 MFG_TOOL 烧写工具->mfgtool 文件夹拷贝到 PC 机（电脑）。

底板拨码开关（BOOT_CFG）设置如下，参考 BOOT_CFG，设置为 USB 连接方式，“1”代码 ON，“0”代表“OFF”。将拨码数字 2 处拨到 ON，其他的为 OFF。如下图

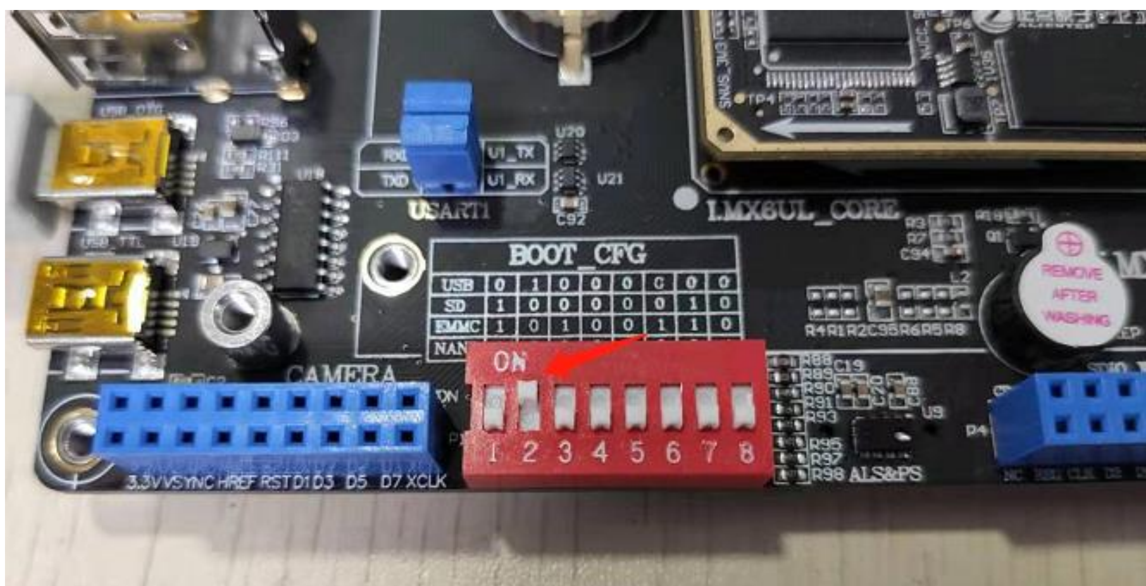


图 2.2.1. 1 mfg 固化系统时底板的拨码开关设置

使用 USB 连接线连接底板的 USB_OTG 接口与 PC 机（电脑）。(注：MINI 底板的位置不一样。)

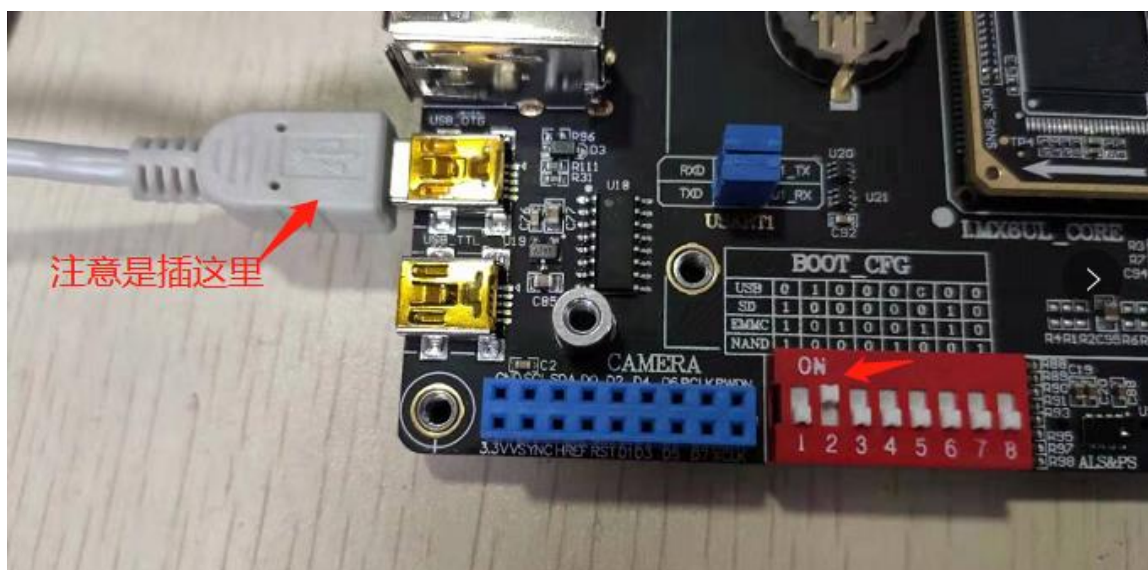


图 2.2.1. 2 用串口线连 USB_OTG 接口

进入 mfgtool 文件夹，查看 vbs 脚本信息，如下图图 2.2.1.1. 1。

名称	修改日期	类型	大小
Drivers	2019/8/26 9:51	文件夹	
Profiles	2019/8/26 9:51	文件夹	
Utils	2019/8/26 9:51	文件夹	
.gitignore	2019/8/26 9:51	文本文档	1 KB
cfg.ini	2019/8/26 9:51	配置设置	1 KB
libMfgToolLib.so	2019/8/26 9:51	SO 文件	6,393 KB
linux-cvbs.sh	2019/8/26 9:51	SH 文件	2 KB
linux-runvbs.sh	2019/8/26 9:51	SH 文件	1 KB
linux-ver-usage	2019/8/26 9:51	文件	1 KB
MfgTool2.exe	2019/8/26 9:51	应用程序	1,955 KB
Mfgtool2-eMMC-ddr256-eMMC.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-eMMC-ddr256-SDCard.vbs	2019/8/29 11:52	VBScript Script ...	1 KB
Mfgtool2-eMMC-ddr512-eMMC.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-eMMC-ddr512-SDCard.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-NAND-ddr256-NAND.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-NAND-ddr256-SDCard.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-NAND-ddr512-NAND.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
Mfgtool2-NAND-ddr512-SDCard.vbs	2019/8/26 9:51	VBScript Script ...	1 KB
mfgtoolcli	2019/8/26 9:51	文件	200 KB
MfgToolLib.dll	2019/8/26 9:51	应用程序扩展	2,192 KB
pax_global_header	2019/8/26 9:51	文件	1 KB
UICfg.ini	2019/8/26 9:51	配置设置	1 KB

图 2.2.1. 3 vbs 脚本信息

这里简单介绍这些 vbs 脚本文件的作用，下图图 2.2.1.1. 2 是这些 vbs 脚本所命名的解释。之所以有那么多不同的 vbs 脚本，是因为正点原子有不同配置的核心板与需要固化的方式不同。我们根据个人的核心板类型不同选择不同的 vbs 脚本固化系统就可以了。

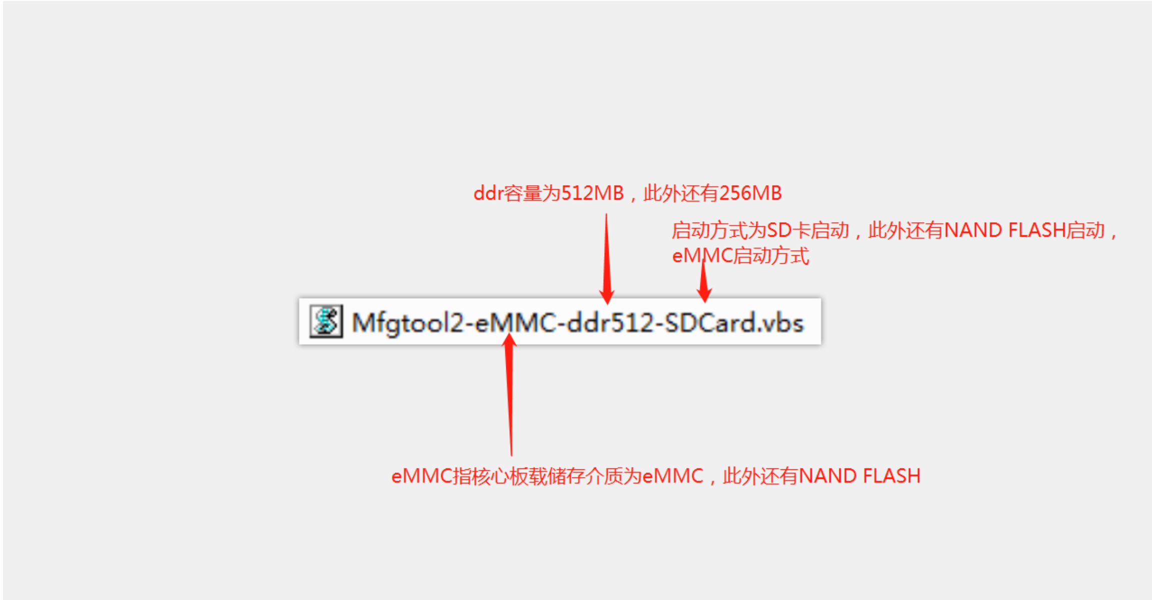


图 2.2.1. 4 vbs 脚本命名解释

2.2.1.1 固化系统到 SD 卡

准备一张 SD 卡（这里用的 TF 小卡），（注意将会格式化 SD 卡，注意备份好重要数据）。

按前面的 vbs 脚本命名解释，比如我们需要从 SD 卡启动，确认核心板存储介质为 eMMC，DDR 容量为 512MB。那么我们就选择双击 Mfgtool2-eMMC-ddr512-SDCard.vbs 进行烧写。（注：

下图图 2.2.1.1. 1 为 windos7 环境提示 NO Device Connected，如果是 windows10 可能是中文的提示。）

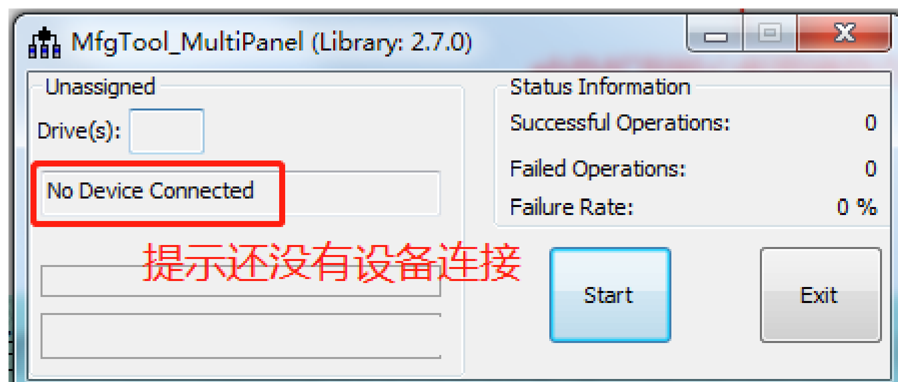


图 2.2.1.1. 1 mfg 上位机提示没有设备连接界面

按上面 mfgtool 使用前准备说明步骤操作后，打上底板电源开关。如果是第一次使用开发板 OTG 连接 PC 机（电脑），需要等待 PC 机自动安装驱动。等待安装驱动完成后，mfgtool 上位机界面会提示已经连接到设备 HID-compliant device，如下图

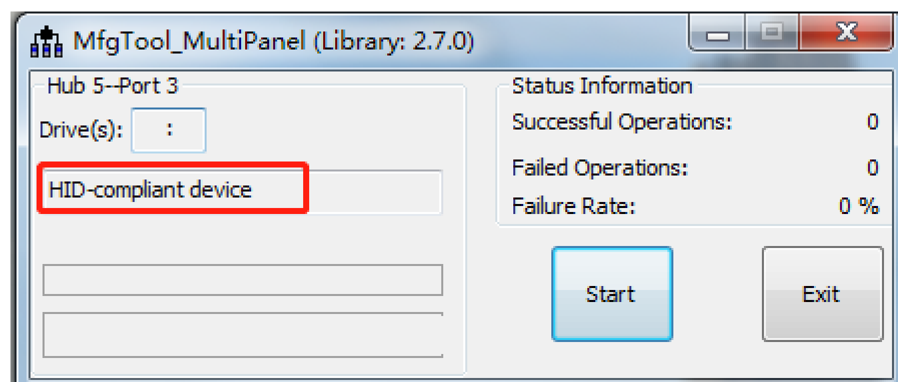


图 2.2.1.1. 2 mfg 上位机连接开发板设备界面

此时将 SD 卡插入卡槽（如果您的 SD 卡原先已经有系统，需开发板先上电后再将 SD 卡插进卡槽，否则上电时会从 SD 卡启动您的系统，这样 mfgtool 会连接不到开发板设备），如下图插上 SD 卡（MINI 底板卡槽的位置不一样）。

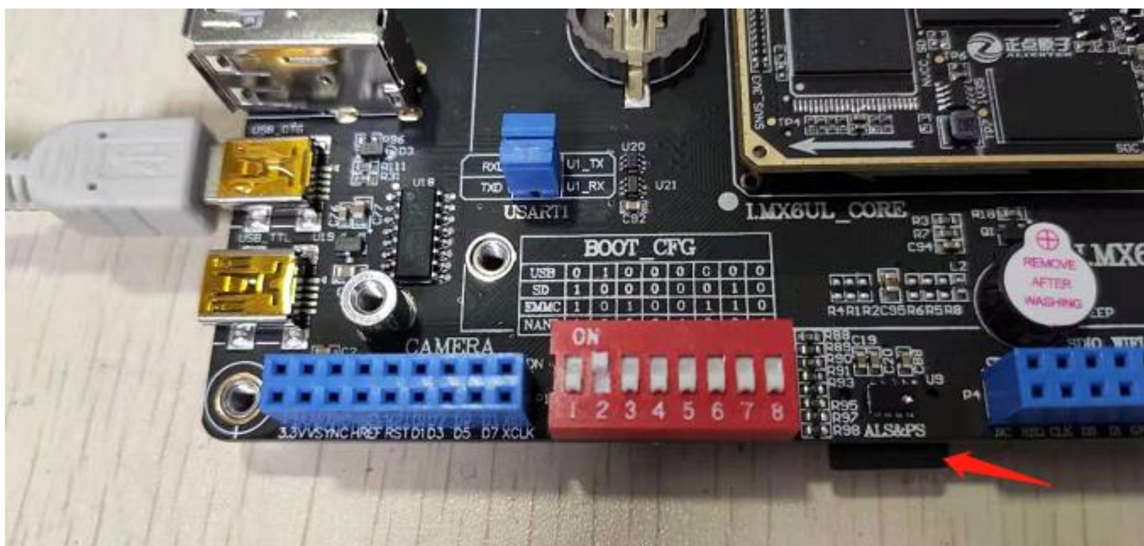


图 2.2.1.1. 3 SD 卡插槽截图

直接点击 mfgtool 的 Start 按钮进行固化系统到 SD 卡。下图为点击 Start 按钮后的截图。固化系统到 SD 卡需要几分钟时间，请耐心等待。

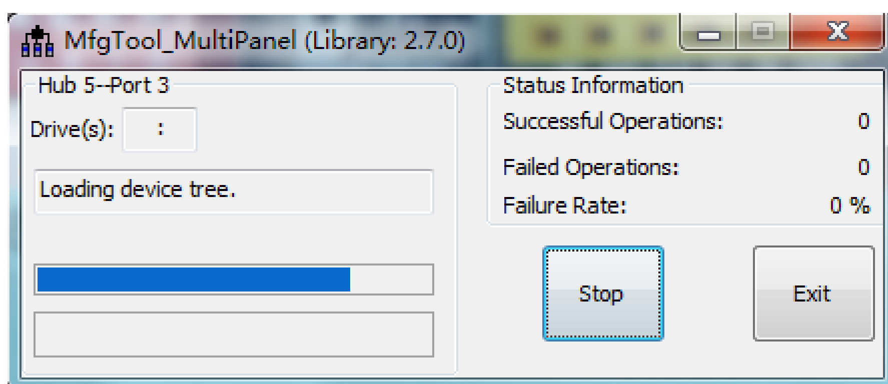


图 2.2.1.1. 4 点击 Start 按钮后截图

下图图 2.2.1.1. 5 为固化过程中的一步，正在写入文件系统

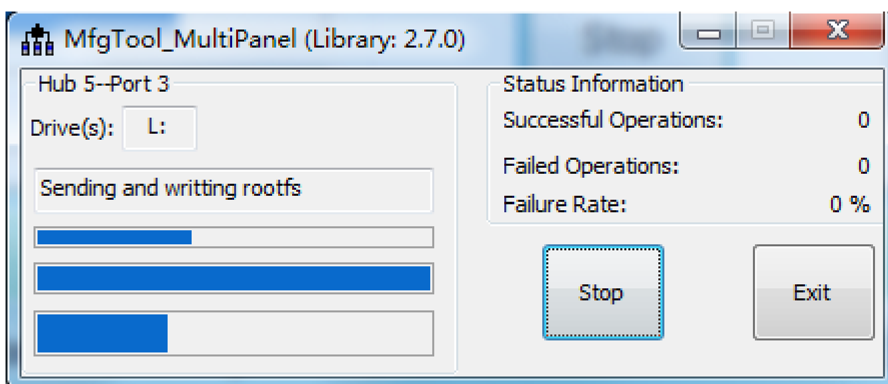


图 2.2.1.1. 5 正在写入文件系统

固化完成如下图，点击 Stop 后再点 Exit 退出 mfgtool 上位机软件即可。

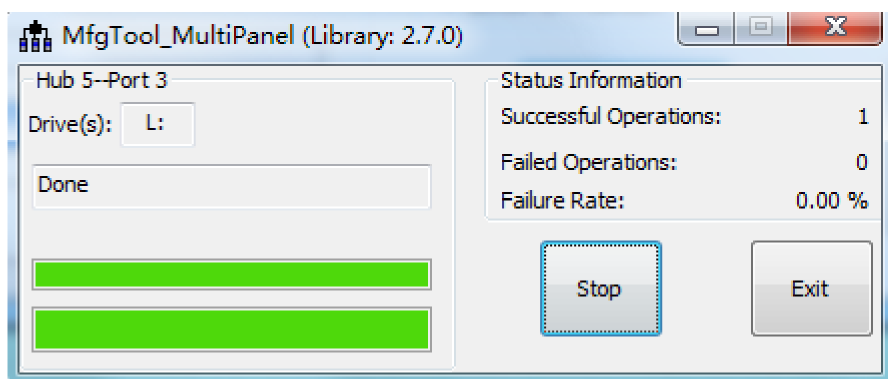


图 2.2.1.1. 6 固化完成

测试从 SD 卡启动系统，拨码开关拨至 SD 卡启动方式 10000010，启动系统即可。

2.2.1.2 固化系统到 eMMC

使用前提：用户核心板类型带 eMMC 存储介质。

请参考固化系统到 SD 卡的步骤（注：固化时不要插入 SD 卡），选择 eMMC 启动方式的 vbs 文件，固化完成后，将拨码开关拨至 eMMC 启动方式 10100110，启动系统即可。

2.2.1.3 固化系统到 NAND FLASH

使用前提：用户核心板类型带 NAND FLASH 存储介质。

请参考固化系统到 SD 卡的步骤（注：固化时不要插入 SD 卡），选择 NAND FLASH 启动方式的 vbs 文件，固化完成后，将拨码开关拨至 NAND 启动方式 10001001，启动系统即可。

2.2.2 使用脚本固化系统

脚本固化系统一般可用于批量固化与升级系统。不像 mfgtool 上位机那样还需要 PC 机和 USB 数据线，用户可以自行修改好固化系统脚本，进行自动化固化测试，那么可以无需专业人员参与，即可批量固化系统。

2.2.2.1 固化系统到 SD 卡

同样地，拷贝->开发板光盘->5、开发工具->4、正点原子修改过的 MFG_TOOL 烧写工具->mfgtool->Profiles->Linux->OS Firmware->files 整个文件夹到 Ubuntu 虚拟机，如下图图 2.2.2.1. 1，本文档已经拷贝 files 文件夹到 Ubuntu 虚拟机。

```
alientek@ubuntu:~/files$ ls
boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules  README.txt
alientek@ubuntu:~/files$
```

图 2.2.2.1. 1 拷贝 files 文件夹到 Ubuntu

使用 chmod 指令修改固化 SD 卡系统脚本 imx6mksdboot.sh 的权限

#Ubuntu chmod +x imx6mksdboot.sh

```
alientek@ubuntu:~/files$ chmod +x imx6mksdboot.sh
alientek@ubuntu:~/files$ ls
boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules  README.txt
alientek@ubuntu:~/files$
```

图 2.2.2.1. 2 赋予脚本可执行权限

将 SD 卡用读卡器插到 Ubuntu 虚拟机，等待 Ubuntu 主机识别后，如下图。（因作者已经制作过 SD 卡，所以 Ubuntu 主机会识别出两个分区，如果您的 SD 卡是空白的，只有一个分区时就识别出一个分区，就会看到只有一个 USB 图标（可能图案不一样））。

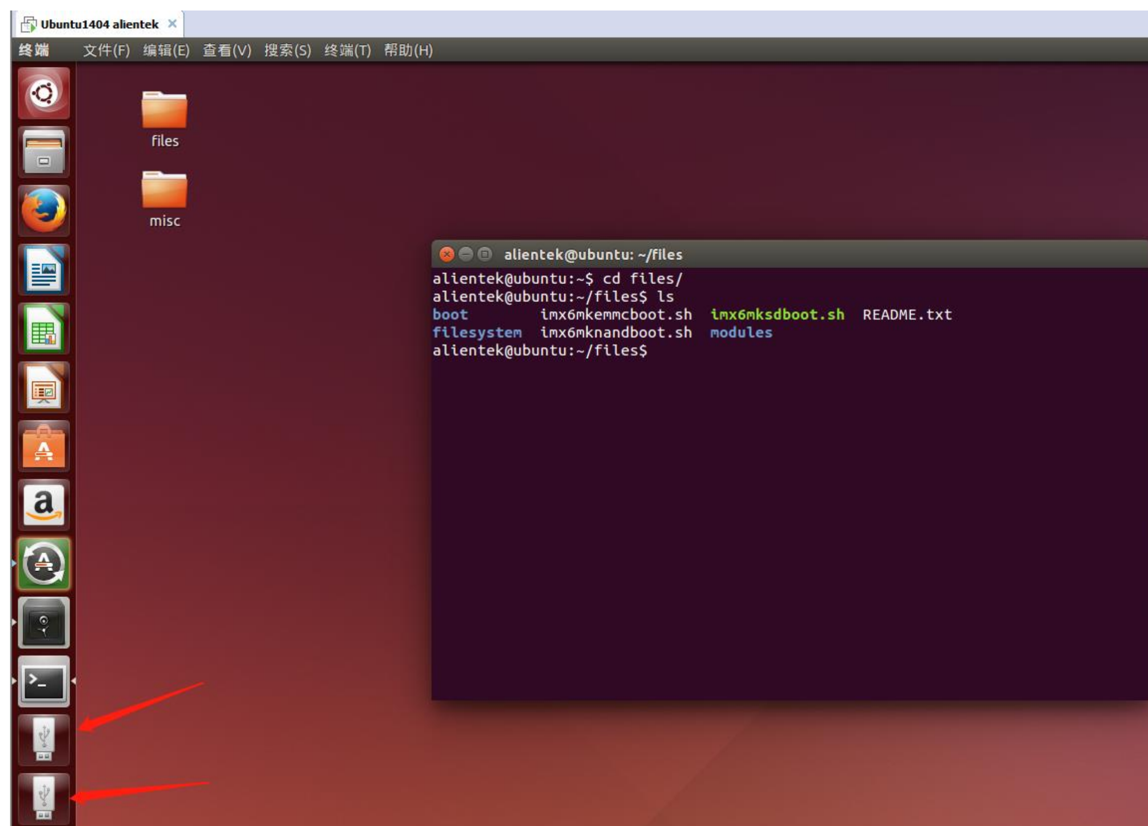


图 2.2.2.1. 3 Ubuntu 识别的 U 盘图标

输入如下指令查看 SD 卡挂载节点，如下图，作者的 SD 卡容量是 8GB 的（7984MB≈8GB），可以看到挂载的节点为/dev/sdb。

#Ubuntu sudo fdisk -l

```
alientek@ubuntu:~/files$ sudo fdisk -l
[sudo] password for alientek:

Disk /dev/sda: 536.9 GB, 536870912000 bytes
255 heads, 63 sectors/track, 65270 cylinders, total 1048576000 sectors
Units = 扇区 of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000028cb

   设备 启动      起点          终点     块数   Id  系统
/dev/sda1   *        2048  1015021567   507509760   83  Linux
/dev/sda2      1015023614  1048573951    16775169    5  扩展
/dev/sda5      1015023616  1048573951    16775168   82  Linux 交换 / Solaris

Disk /dev/sdb: 7984 MB, 7984906240 bytes
246 heads, 62 sectors/track, 1022 cylinders, total 15595520 sectors
Units = 扇区 of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x38e748af

   设备 启动      起点          终点     块数   Id  系统
/dev/sdb1            2048      264191      131072    c  W95 FAT32 (LBA)
/dev/sdb2          264192    15595519     7665664   83  Linux
alientek@ubuntu:~/files$
```

图 2.2.2.1. 4 查看 U 盘在 Ubuntu 上的连接节点

直接执行./imx6mksdboot.sh --help 查看脚本的使用方法

```
alientek@ubuntu:~/files$ ./imx6mksdboot.sh --help


用法: imx6mksdboot.sh [选项] <(必选)-device> <(可选)-flash> <(可选)-ddrsize>
用法示例：
sudo ./imx6mksdboot.sh -device /dev/sdd
sudo ./imx6mksdboot.sh -device /dev/sdd -flash emmc -ddrsize 512
命令选项:
  -device              SD卡块设备节点 (例如/dev/sdx)
  -flash               请选择开发板Flash类型 ( emmc | nand )
  -ddrsize             请选择DDR大小 ( 512 | 256 )
可选选项:
  --version             打印版本信息.
  --help                打印帮助信息.

alientek@ubuntu:~/files$
```

图 2.2.2.1. 5 查看 imx6mksdboot.sh 使用说明

用法说明：

用法: imx6mksdboot.sh [选项] <(必选)-device> <(可选)-flash> <(可选)-ddrsize>

-device：指明设备节点（SD 卡挂载的节点如/dev/sdx），执行脚本时必需要加这个参数

-flash：指明核心板上的媒体存储介质，可选为（emmc|nand）

-ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

比如现在用户是核心板的 ddr 容量大小是 512MB，媒体存储介质是 eMMC。SD 卡挂载节点为/dev/sdb。那么固化 SD 卡的指令如下，执行指令后脚本会有中文再次询问 SD 卡所挂载的节点是否对应，将会清空 SD 卡上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化 SD 卡需要大约需要几分钟时间，这里根据个人电脑不同和所用 SD 卡不同，可能花费的时间差异比较大。

#Ubuntu sudo ./imx6mksdboot.sh -device /dev/sdb -flash emmc -ddrsize 512

```
alientek@ubuntu:~/files$ sudo ./imx6mksdboot.sh -device /dev/sdb -flash emmc -ddrsize 512
[sudo] password for alientek:
您已经选择开发板参数为：EMMC版本，DDR大小为512MB
即将进行制作SD系统启动卡，大约花费几分钟时间,请耐心等待!
************************************************************
*         注意：这将会清除/dev/sdb所有的数据               *
*         在脚本执行时请不要将/dev/sdb拔出                 *
*             请按<Enter>确认继续                          *
************************************************************
```

图 2.2.2.1. 6 执行固化系统到 SD 卡

固化时有中英文结合，提示固化的过程，固化完成如下图

```
命令(输入 m 获取帮助)： The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
格式化 /dev/sdb1 ...
mkfs.fat 3.0.26 (2014-03-07)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
格式化/dev/sdb2 ...
mke2fs 1.42.9 (4-Feb-2014)
文件系统标签=rootfs
OS type: Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
483328 inodes, 1932800 blocks
96640 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=1979711488
59 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: 完成
正在写入inode表: 完成
Creating journal (32768 blocks): 完成
Writing superblocks and filesystem accounting information: 完成

正在烧写u-boot-imx6ull-14x14-ddr512-emmc.imx到/dev/sdb
记录了415+0 的读入
记录了415+0 的写出
424960字节(425 kB)已复制，0.693281 秒，613 kB/秒
烧写u-boot-imx6ull-14x14-ddr512-emmc.imx到/dev/sdb完成！
正在准备复制...
正在复制设备树与内核到/dev/sdb1，请稍候...
复制设备树与内核到/dev/sdb1完成！
卸载/dev/sdb1
正在解压文件系统到/dev/sdb2 ，请稍候...
解压文件系统到/dev/sdb2完成！
正在解压模块到/dev/sdb2/lib/modules/ ，请稍候...
解压模块到/dev/sdb2/lib/modules/完成！
卸载/dev/sdb2
SD卡启动系统烧写完成！
alientek@ubuntu:~/files$
```

图 2.2.2.1. 7 固化完成截图

完成固化后，将 SD 卡卸载后再取出，单击 USB 图标，点击“2”处小三角卸载退出后直接拔出读卡器。

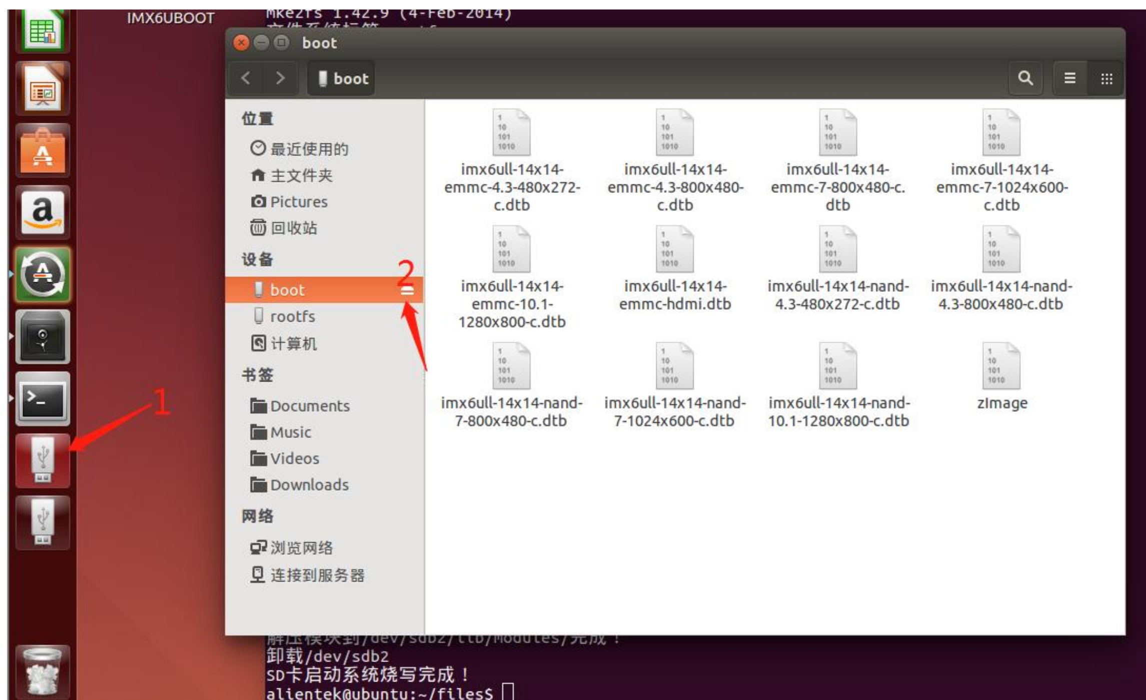


图 2.2.2.1. 8 卸载 SD 卡分区

测试从 SD 卡启动系统，拨码开关拨至 SD 卡启动方式 10000010，启动系统即可。

2.2.2.2 固化系统到 eMMC

使用前提：用户核心板类型带 eMMC 存储介质

这里使用脚本固化系统到 eMMC 需要使用 2.1 制作好的 SD 系统启动卡。同样的，拷贝->开发板光盘->5、开发工具->4、正点原子修改过的 MFG_TOOL 烧写工具->mfgtool->Profiles->Linux->OS Firmware->files 文件夹到制作好的 SD 系统启动卡里面的/home/root 目录（本文拷贝到/home/root 目录，读者可任意目录）。

```
root@ALIENTEK:~/files# ls
README.txt  boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules
root@ALIENTEK:~/files# pwd
/home/root/files
root@ALIENTEK:~/files#
```

图 2.2.2.2. 1 拷贝 files 文件夹到开发板 SD 启动卡文件系统中

修改 eMMC 固化脚本的权限

USER# chmod +x im6mkemmcboot.sh

```
root@ALIENTEK:~/files# chmod +x imx6mkemmcboot.sh
root@ALIENTEK:~/files#
```

图 2.2.2.2. 2 赋予 im6mkemmcboot.sh 可执行权限

执行./imx6mkemmcboot.sh --help 查看脚本的使用说明

```
root@ALIENTEK:~/files# ./imx6mkemmcboot.sh --help

用法: imx6mkemmcboot.sh [选项] <(必选)-device> <(可选)-ddrsize>
用法示例:
./imx6mkemmcboot.sh -device /dev/mmcblk1
./imx6mkemmcboot.sh -device /dev/mmcblk1 -ddrsize 512
命令选项:
  -device                 eMMC块设备节点 (一般eMMC设备节点为/dev/mmcblk1)
  -ddrsize                请选择DDR大小 (512 | 256)
可选选项:
  --version                打印版本信息.
  --help                   打印帮助信息.

root@ALIENTEK:~/files#
```

图 2.2.2.2. 3 查看 imx6mkemmcboot.sh 的脚本使用说明

使用 fdisk 指令查看 eMMC 挂载节点，一般挂载节点为/dev/mmcblk1，本文测试的 eMMC 为 8GB 存储容量的。可以看到下图/dev/mmcblk1 就是 eMMC 的挂载节点。

USER# fdisk -l

```
Device         Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk0p1       20480   282623   262144  128M  c W95 FAT32 (LBA)
/dev/mmcblk0p2      282624 15613951 15331328  7.3G 83 Linux

Disk /dev/mmcblk1: 7.3 GiB, 7818182656 bytes, 15269888 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x7848174f

Device         Boot Start      End  Sectors  Size Id Type
/dev/mmcblk1p1 *     2048    67583    65536   32M  c W95 FAT32 (LBA)
/dev/mmcblk1p2      67584 15269887 15202304  7.3G 83 Linux


Disk /dev/mmcblk1boot1: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk1boot0: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@ALIENTEK:~/files#
```

图 2.2.2.2. 4 查看 eMMC 挂载节点

用法说明：

用法: imx6mkemmcboot.sh [选项] <(必选)-device> <(可选)-ddrsize>

-device：指明设备节点（eMMC 挂载的节点如/dev/mmcblk1），执行脚本时必需要加这个参数

-ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

比如现在用户是核心板的 ddr 容量大小是 512MB，eMMC 挂载节点为/dev/mmcblk1。那么固化的指令如下，执行指令后脚本会有中文再次询问 eMMC 所挂载的节点是否对应，将会清空 eMMC 上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化系统到 eMMC 需要大约需要几分钟时间。

USER# ./imx6mkemmcboot.sh -device /dev/mmcblk1 -ddrsize 512

```
root@ALIENTEK:~/files# ./imx6mkemmcboot.sh -device /dev/mmcblk1 -ddrsize 512
您已经选择开发板参数为：DDR大小为512MB
即将进行制作eMMC系统启动卡，大约花费几分钟时间,请耐心等待!
************************************************************************
*            注意：这将会清除/dev/mmcblk1所有的数据                      *
             烧写eMMC前，请备份好重要的数据                              *
*              请按<Enter>确认继续                                       *
************************************************************************
```

图 2.2.2.2. 5 执行固化脚本指令

固化系统完成如下图

```
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x12ad64bc.

Command (m for help): Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): Partition number (1-4, default 1): First sector (2048-15269887, default 2048): Last sector, +sectors or +size{K,M,G,T,P} (2048-15269887, default 15269887):
Created a new [ 2546.412684]  mmcblk1: p1 p2
partition 1 of type 'Linux' and of size 32 MiB.

Command (m for help): Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): Partition number (2-4, default 2): First sector (67584-15269887, default 67584): Last sector, +sectors or +size{K,M,G,T,P} (67584-15269887, default 15269887):
Created a new partition 2 of type 'Linux' and of size 7.3 GiB.

Command (m for help): Partition number (1,2, default 2): Partition type (type L to list all types):
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): Partition number (1,2, default 2):
The bootable flag on partition 1 is enabled now.

Command (m for help): The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[ 2547.739173] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[ 2547.768979] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
卸载 device '/dev/mmcblk1p1'
卸载 device '/dev/mmcblk1p2'
Formating /dev/mmcblk1p1 ...
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
WARNING: Not enough clusters for a 32 bit FAT!
格式化/dev/mmcblk1p2 ...
mke2fs 1.43-WIP (18-May-2015)
/dev/mmcblk1p2 contains a ext4 file system labelled 'rootfs'
        last mounted on / on Wed Aug 14 13:26:00 2019
Discarding device blocks: done
Creating filesystem with 1900288 4k blocks and 475136 inodes
Filesystem UUID: edaaee80-9339-4536-82d1-4f374191e127
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

正在烧写u-boot-imx6ull-14x14-ddr512-emmc.imx到/dev/mmcblk1
415+0 records in
415+0 records out
424960 bytes (425 kB, 415 KiB) copied, 0.105774 s, 4.0 MB/s
415+0 records in
415+0 records out
424960 bytes (425 kB, 415 KiB) copied, 0.0895603 s, 4.7 MB/s
正在准备复制...
正在复制设备树与内核到/dev/mmcblk1p1, 请稍候...
复制设备树与内核到/dev/mmcblk1p1完成!
卸载/dev/mmcblk1p1
[ 2588.453093] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
正在解压文件系统到/dev/mmcblk1p2 , 请稍候...
解压文件系统到/dev/mmcblk1p2完成!
正在解压模块到/dev/mmcblk1p2/lib/modules/, 请稍候...
解压模块到/dev/mmcblk1p2/lib/modules/完成!
卸载/dev/mmcblk1p2
eMMC启动系统烧写完成!
root@ALIENTEK:~/files#
```

图 2.2.2.2. 6 固化系统到 eMMC 完成

固化完成后，将拨码开关拨至 eMMC 启动方式 10100110，启动系统即可。

2.2.2.3 固化系统到 NAND FLASH

使用前提：用户核心板类型带 Nand Flash 存储介质

这里使用脚本固化系统到 Nand Flash 需要使用 2.1 制作好的 SD 系统启动卡。同样地，拷贝->开发板光盘->5、开发工具->4、正点原子修改过的 MFG_TOOL 烧写工具->mfgtool->Profiles->Linux->OS Firmware->files 文件夹到制作好的 SD 系统启动卡里面的/home/root 目录（本文拷贝到/home/root 目录，读者可任意目录）。

```
root@ALIENTEK-IMX6U:~/files# pwd
/home/root/files
root@ALIENTEK-IMX6U:~/files# ls
README.txt  boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules
root@ALIENTEK-IMX6U:~/files#
```

图 2.2.2.3. 1 拷贝 files 文件夹到开发板 SD 启动卡文件系统中

使用 cat 指令查看 MTD 分区表，打印结果如下图表示存在 Nand Flash。

USER# cat /proc/mtd

```
root@ALIENTEK-IMX6U:~/files# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00020000 "u-boot"
mtd1: 00020000 00020000 "env"
mtd2: 00100000 00020000 "logo"
mtd3: 00100000 00020000 "dtb"
mtd4: 00800000 00020000 "kernel"
mtd5: 1f1e0000 00020000 "rootfs"
root@ALIENTEK-IMX6U:~/files#
```

图 2.2.2.3. 2 查看 Nand Flash 的分区信息

修改 NandFlash 固化脚本的权限

USER# chmod +x imx6mknandboot.sh

```
root@ALIENTEK-IMX6U:~/files# chmod +x imx6mknandboot.sh
root@ALIENTEK-IMX6U:~/files#
```

图 2.2.2.3. 3 赋予 imx6mknandboot.sh 可执行权限

执行./imx6mknandboot.sh --help 查看脚本的使用说明

```
root@ALIENTEK-IMX6U:~/files# ./imx6mknandboot.sh --help


用法: imx6mknandboot.sh [选项] <(可选)-ddrsize>
用法示例:
./imx6mknandboot.sh
./imx6mknandboot.sh -ddrsize 512
命令选项:
  -ddrsize              请选择DDR大小  (512 | 256)
可选选项:
  --version              打印版本信息.
  --help                 打印帮助信息.

root@ALIENTEK-IMX6U:~/files#
```

图 2.2.2.3. 4 查看 imx6mknandboot.sh 使用说明

用法说明：

用法:imx6mknandboot.sh [选项] <(可选)-ddrsize>

-ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

比如现在用户是核心板的 ddr 容量大小是 256MB，那么固化的指令如下。这将会清空 NandFlash 上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化系统到 NandFlash 需要大约需要几分钟时间。

USER# ./imx6mknandboot.sh -ddrsize 256

```
root@ALIENTEK-IMX6U:~/files# ./imx6mknandboot.sh -ddrsize 256
您已经选择开发板参数为: DDR大小为256MB
即将进行固化NandFlash系统, 大约花费几分钟时间,请耐心等待!
*****************************************************************
*          注意: 这将会清除NandFlash所有的数据                 *
*        在脚本执行时请勿断电或者中断烧写过程                   *
*             请按<Enter>确认继续                               *
*****************************************************************
```

图 2.2.2.3. 5 执行固化系统到 Nand Flash 指令

固化系统完成如下图图 2.2.2.3. 6。

```
root@ALIENTEK-IMX6U:~/files# ./imx6mknandboot.sh -ddrsize 256
您已经选择开发板参数为：DDR大小为256MB
即将进行固化NandFlash系统，大约花费几分钟时间,请耐心等待!
************************************************************
*        注意：这将会清除NandFlash所有的数据                  *
*      在脚本执行时请勿断电或者中断烧写过程                   *
*             请按<Enter>确认继续                             *
************************************************************

正在擦除Uboot分区...
Erasing 128 Kibyte @ 3e0000 -- 100 % complete
正在烧写U-boot...
Erasing 128 Kibyte @ 0 -- 100 % complete
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在擦除设备树分区...
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在烧写设备树...
正在擦除内核分区...
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
正在烧写内核...
正在擦除文件系统分区，请稍候...
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
ubiformat: mtd5 (nand), size 522059776 bytes (497.9 MiB), 3983 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size 2048 bytes
libscan: scanning eraseblock 3982 -- 100 % complete
ubiformat: 3979 eraseblocks are supposedly empty
ubiformat: 4 bad eraseblocks found, numbers: 3979, 3980, 3981, 3982
ubifor[  582.862209] ubi0: attaching mtd599 % complete
ubiformat: formatting eraseblock 3982 -- 100 % complete
[  587.473859] ubi0: scanning is finished
[  587.505852] ubi0: attached mtd5 (name "rootfs", size 497 MiB)
[  587.511627] ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
[  587.521800] ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
[  587.528844] ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
[  587.536752] ubi0: good PEBs: 3979, bad PEBs: 4, corrupted PEBs: 0
[  587.543041] ubi0: user volume: 0, internal volumes: 1, max. volumes count: 128
[  587.550278] ubi0: max/mean erase counter: 0/0, WL threshold: 4096, image sequence number: 580955628
[  587.560239] ubi0: available PEBs: 3899, total reserved PEBs: 80, PEBs reserved for bad PEB handling: 76
[  587.569819] ubi0: background thread "ubi_bgt0d" started, PID 718
UBI device number 0, total 3979 LEBs (505237504 bytes, 481.8 MiB), available 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (124.0 KiB)
Set volume size to 495079424
Volume ID 0, size 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (124.0 KiB), dynamic, name "rootfs", alignment 1
[  587.671655] UBIFS (ubi0:0): default file-system created
[  587.678606] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" started, PID 722
[  587.799865] UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name "rootfs"
[  587.807542] UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O unit sizes: 2048 bytes/2048 bytes
[  587.818744] UBIFS (ubi0:0): FS size: 493047808 bytes (470 MiB, 3883 LEBs), journal size 24633344 bytes (23 MiB, 194 LEBs)
[  587.829875] UBIFS (ubi0:0): reserved for root: 4952683 bytes (4836 KiB)
[  587.837354] UBIFS (ubi0:0): media format: w4/r0 (latest is w4/r0), UUID 593A9318-16E9-4FCC-8AE6-275299FAFEDE, small LPT model
正在解压文件系统到mtd5分区，请稍候...
正在解压模块到mtd5分区，请稍候...
tar (child): /home/root/files/moudules/*.tar.*: Cannot open: No such file or directory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
[  931.397155] UBIFS (ubi0:0): un-mount UBI device 0
[  931.401944] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" stops
[  931.494436] ubi0: detaching mtd5
[  931.505214] ubi0: mtd5 is detached
NandFlash启动系统烧写完成!
root@ALIENTEK-IMX6U:~/files#
```

图 2.2.2.3. 6 固化完成

固化完成后，将拨码开关拨至 Nand Flash 启动方式 10001001，启动系统即可。

2.3 登录开发板

2.3.1 拨码开关设置

请根据个人核心板上的存储介质类型，选择不同的方式启动动系统。如下图参照底板原理图说明：

SW

D1	D2	D3	D4	D5	D6	D7	D8	BOOT DEVICE
OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	USB
ON	OFF	OFF	OFF	OFF	OFF	ON	OFF	MicroSD
ON	OFF	ON	OFF	OFF	ON	ON	OFF	EMMC
ON	OFF	OFF	OFF	ON	OFF	OFF	ON	NAND

图 2.3.1. 1 底板原理图拨码开关设置

解释：OFF 为 0，ON 为 1。

- USB OTG 烧写设置：0100 0000
- SD 卡启动设置：1000 0010
- EMMC 启动设置：1010 0110
- NAND FLASH 启动设置：1000 1001

用户选择正确的启动方式后，开发板上电，进入系统后，开发板自动登录，无需输入用户名及密码。

第三章 ALIENTEK I.MX6U 功能测试

3.1.LED 与蜂鸣器测试

说明：用户 LED 一个，蜂鸣器一个。开发板启动时，DS0 作为心跳灯，用于指示系统的运行

开发板与 LED 和蜂鸣器对应的管脚关系如下：

开发板	GPIO03	SNVS_TAMPER1
ALPHA	DS0	BEEP

进入开发板系统，在串口终端执行指令控制对应的 IO 来控制对应的器件：

开发板上启动后 DS0 默认是[heartbeat]模式，执行如下指令改变当前触发模式，改成[none]模式就可以通过指令来控制 LED 的亮灭了。

USER# echo none > /sys/class/leds/sys-led/trigger　　　　// 改变 LED 的触发模式

USER# echo 1 > /sys/class/leds/sys-led/brightness　　　　// 点亮 LED

USER# echo 0 > /sys/class/leds/sys-led/brightness　　　　// 熄灭 LED

```
root@ALIENTEK-IMX6:~# echo none > /sys/class/leds/sys-led/trigger
root@ALIENTEK-IMX6:~# echo 1 > /sys/class/leds/sys-led/brightness
root@ALIENTEK-IMX6:~# echo 0 > /sys/class/leds/sys-led/brightness
root@ALIENTEK-IMX6:~#
```

图 3.1. 1 输入指令控制 led

因为底板的蜂鸣器用配置成了 gpio-leds 模式，同理蜂鸣器也可以用这样的指令来控制，默认 BEEP 的触发模式为[none]。

USER# echo 1 > /sys/class/leds/beep/brightness　　　　// 鸣叫

USER# echo 0 > /sys/class/leds/beep/brightness　　　　// 关闭

```
root@ALIENTEK-IMX6:~# echo 1 > /sys/class/leds/beep/brightness
root@ALIENTEK-IMX6:~# echo 0 > /sys/class/leds/beep/brightness
root@ALIENTEK-IMX6:~#
```

图 3.1. 2 输入指令控制 beep

3.2 按键测试

底板上按键对应的管脚关系如下：

开发板	GPIO18
ALPHA	KEY0

进入开发板系统，在串口终端执行如下指令查看按键所对应的输入事件

USER# lsinput

```
root@ALIENTEK-IMX6:~# lsinput
/dev/input/event0
   bustype : BUS_HOST
   vendor  : 0x0
   product : 0x0
   version : 0
   name    : "20cc000.snvs:snvs-powerkey"
   phys    : "snvs-pwrkey/input0"
   bits ev : EV_SYN EV_KEY

/dev/input/event1
   bustype : BUS_I2C
   vendor  : 0xdead
   product : 0xbeef
   version : 10427
   name    : "goodix-ts"
   phys    : "input/ts"
   bits ev : EV_SYN EV_KEY EV_ABS

/dev/input/event2
   bustype : BUS_HOST
   vendor  : 0x1
   product : 0x1
   version : 256
   name    : "gpio_keys@0"
   phys    : "gpio-keys/input0"
   bits ev : EV_SYN EV_KEY EV_REP

open /dev/input/event3: No such file or directory
root@ALIENTEK-IMX6:~#
```

图 3.2.1 查看按键的输入事件

可以从上图看出按键事件号为 event2，触摸屏占用了 event1，所以当触摸屏没有插上时按键事件号为不一定为 event2！执行下面的指令，进行按键测试，按下底板上的 KEY0，打印出按键输入事件的信息如下，按“Ctrl+c”终止指令。

指令提示：hexdump 或者 od -x 指令都是以十六进制的形式打印出输入事件信息。由于文件系统没提供 hexpdump 指令，所以只测试 od 指令。

USER# od -x /dev/input/event2

```
root@ALIENTEK-IMX6:~# od -x /dev/input/event2
0000000 7caf 5cf6 cd10 0008 0001 0072 0001 0000
0000020 7caf 5cf6 cd10 0008 0000 0000 0000 0000
0000040 7caf 5cf6 da46 000b 0001 0072 0000 0000
0000060 7caf 5cf6 da46 000b 0000 0000 0000 0000
0000100 7cb0 5cf6 1a6f 0000 0001 0072 0001 0000
0000120 7cb0 5cf6 1a6f 0000 0000 0000 0000 0000
0000140 7cb0 5cf6 8b64 0002 0001 0072 0000 0000
0000160 7cb0 5cf6 8b64 0002 0000 0000 0000 0000
^C
root@ALIENTEK-IMX6:~#
```

图 3.2. 2 打印按键输入事件的信息

3.3 LCD 触摸屏

ALPHA 开发板已经兼容所有本公司的 RGB 屏幕，进行 LCD 触摸屏实验时请先断电插上 RGB 屏幕，再启动系统。

3.3.1 LCD 背光调节

LCD 屏幕的背光支持 8 级变化，亮度级数为 0～7，默认为 6。

查看 LCD 屏幕最大亮度等级

USER# cat /sys/devices/platform/backlight/backlight/backlight/max_brightness

```
root@ALIENTEK-IMX6:~# cat /sys/devices/platform/backlight/backlight/backlight/max_brightness
7
root@ALIENTEK-IMX6:~#
```

图 3.3.1. 1 查看 LCD 屏幕背光最大亮度等级

查看当前 LCD 屏幕背光亮度等级

USER# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness

```
root@ALIENTEK-IMX6:~# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
6
root@ALIENTEK-IMX6:~#
```

图 3.3.1. 2 查看当前 LCD 屏幕背光亮度等级

修改当前 LCD 屏幕背光亮度等级，修改后再查看

USER# echo 5 > /sys/class/backlight/backlight/brightness

USER# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness

```
root@ALIENTEK-IMX6:~# echo 5 > /sys/class/backlight/backlight/brightness
root@ALIENTEK-IMX6:~# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
5
root@ALIENTEK-IMX6:~#
```

图 3.3.1. 3 查看修改后的 LCD 屏幕背光亮度等级

3.3.2 LCD 触摸校准

先退出 Qt 桌面（若使用不含 Qt 的文件系统不用退出）。

直接执行下面的指令可退出桌面程序。psplash.sh 这个脚本会关闭以 Q 开头的 Qt 程序。关闭桌面程序后，可执行下面的指令进行重启桌面程序/opt/qt5.5.1/apps/QDesktop/QDesktop &

USER# /etc/init.d/psplash.sh

触摸校准，以下实验是对本公司 7 寸屏 800*480 分辨率的屏幕进行校准测试。

USER# ts_calibrate

```
root@ALIENTEK-IMX6U:~# ts_calibrate
xres = 800, yres = 480
Took 24 samples...
Top left : X =   49 Y =   54
Took 21 samples...
Top right : X =   44 Y =  739
Took 22 samples...
Bot right : X =  443 Y =  742
Took 24 samples...
Bot left : X =  440 Y =   61
Took 16 samples...
Center : X =  250 Y =  400
-5.949463 -0.012948 1.024861
3.618195 0.961732 0.001416
Calibration constants: -389904 -848 67165 237122 63028 92 65536
root@ALIENTEK-IMX6U:~#
```

LCD 屏幕显示校准界面如下图



执行命令后 LCD 会弹出校准界面，请依次点击校准准星。连续点击五次之后，会在"/etc"目录下生成触摸屏校准文件 pointercal，校准后的信息记录在 pointercal 文件中。

备注：因 Qt 触摸使用插件 Tslib，需要参考 pointercal 文件参数作为触摸坐标参考。这里我们文件系统里已经为用户准备了好几份 pointercal 参数，在/opt/qt5.5.1/apps/pointercals/目录下。auto-cover-pointercal.sh 脚本在开机时候会自动执行，它的作用是判断用户 RGB 屏的参数，从而匹配一个 pointercal 文件复制到/etc 目录下，这样用户拿到板子就不用校准就可以使用了。若用户需要使用自己校准的 pointercal 文件，那么将校准之后的 pointercal 重命名为下图对应屏幕参数之一，替换掉下图对应屏幕参数的 pointercal。

```
root@ALIENTEK-IMX6U:~# ls /opt/qt5.5.1/apps/pointercals/
auto-cover-pointercal.sh  pointercal-10.1-1280x800  pointercal-4.3-480x272  pointercal-4.3-800x480  pointercal-7-1024x600  pointercal-7-800x480
root@ALIENTEK-IMX6U:~#
root@ALIENTEK-IMX6U:~#
root@ALIENTEK-IMX6U:~#
```

3.3.3 LCD 显示控制

进入/退出睡眠模式，注：这里指控制 LCD 进入睡眠模式（实质是操控 fb0），并不是整机进入睡眠模式。

进入睡眠/熄屏模式：

USER#　echo "4" > /sys/class/graphics/fb0/blank

```
root@ALIENTEK-IMX6U:~# echo "4" > /sys/class/graphics/fb0/blank
```

退出睡眠/亮屏模式：

USER#　echo "0" > /sys/class/graphics/fb0/blank

```
root@ALIENTEK-IMX6U:~# echo "0" > /sys/class/graphics/fb0/blank
```

3.4 串口测试

3.4.1 RS232 串口测试

测试前准备正点原子 USB 转换器和 RS232 转接头（或者直接用 USB 转 RS232 串口线）。

如下图，下图准备的是正点原子的 USB 转换器。可以测试 RS232 与 RS485 等。（注：3.4.2 小节测试 RS485 也是用这个模块。）



图 3.4.1. 1 正点原子 USB 转换器模块

底板 RS232 接口与 RS485 接口的 Tx 与 Rx 是共用的，使用时需要使用跳线帽进行切换。RS232 串口线的一头母头接到开发板底板的 COM3 接口处，另一头连接 USB 转换器再连接计算机。测试前请将跳帽将 U3_Tx 与 COM3_Rx 连接，U3_Rx 与 COM3_Tx 连接。切换跳线帽的位置如下图图 3.4.1. 2。

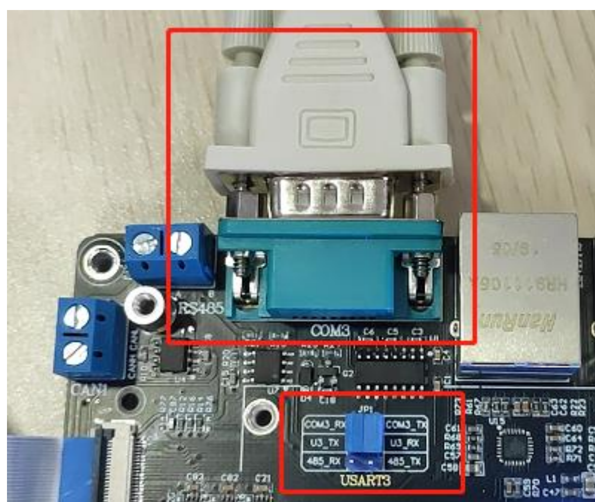


图 3.4.1. 2 RS232 接口与切换跳线帽的位置

在计算机的设备查看 USB 转换器的端口号。作者的端口号有两个，一个是开发板 USB 调试串口的，另一个就是 USB 转换器的端口号了。

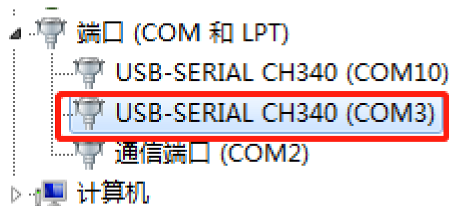


图 3.4.1. 3 在设备管理器查看 USB 转换器的端口号

选择正确的 COM 口，波特率为 115200，8N1，无检验位，并建立串口连接。

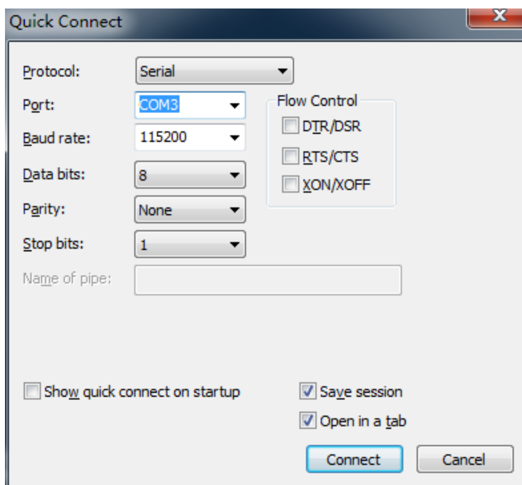


图 3.4.1.4 串口调试终端设置

下图 serial-com10 为 USB 调试串口的终端，serial-com3 是 usb 转换器的终端

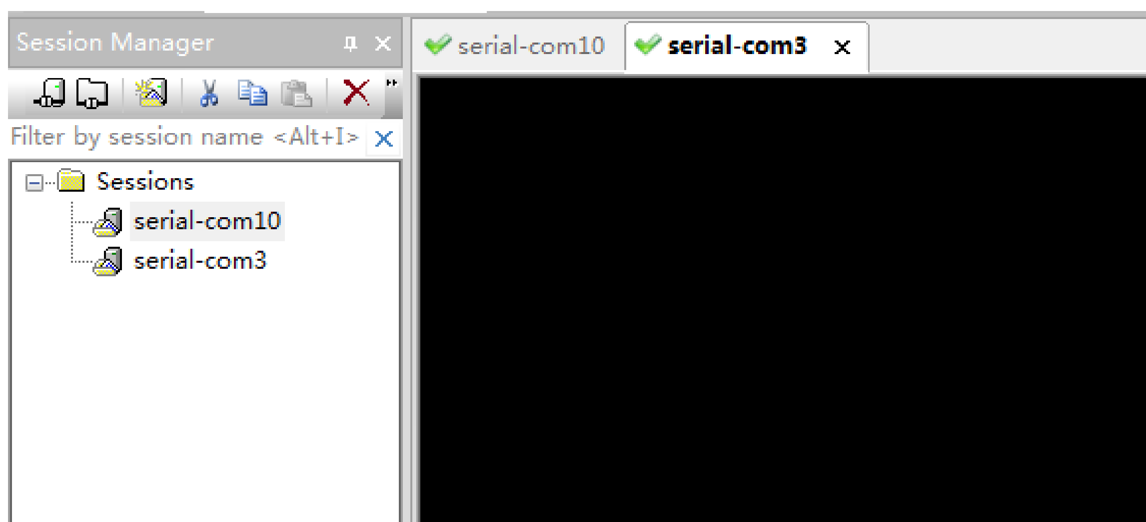


图 3.4.1.5 连接的 RS232 串口 COM3

在 usb 串口调试终端 COM10 输入下面的指令，将 RS232 也设置成一个串口终端。

USER# setsid getty 115200 /dev/ttymxc2

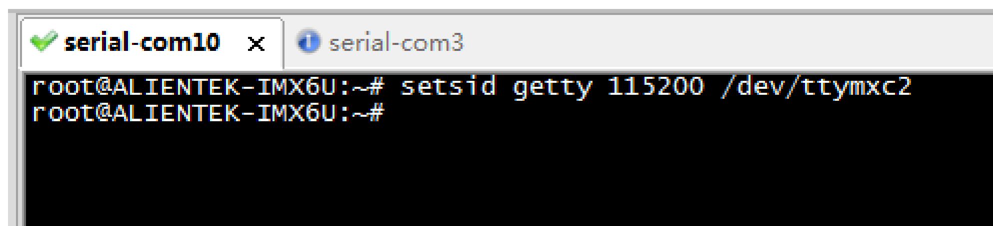


图 3.4.1.6 使用指令设置 RS232 为一个串口终端

这样可以像调试串口一样输入登录用户名 root，即可进入系统。这样能输入指令并返回结果，表明 RS232 串口正常。

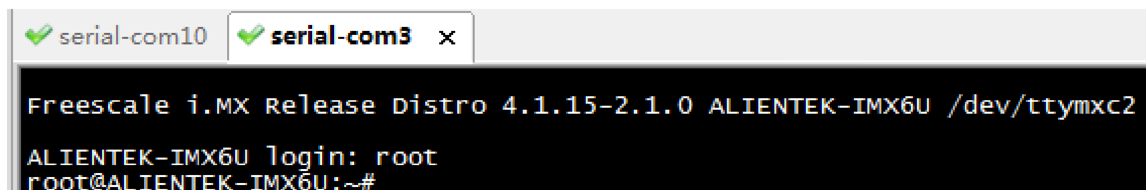


图 3.4.1.7 输入“root”指令登录开发板

3.4.2 RS485 串口测试

与 RS232 测试方法一样，使用 USB 转换器测试 RS485 接口（或者用户手上有其他 RS232 转 RS485 的模块亦可）。测试前请将跳帽将 U3_Tx 与 485_Rx 连接，U3_Rx 与 485_Tx 连接。切换跳线帽的位置如下图。

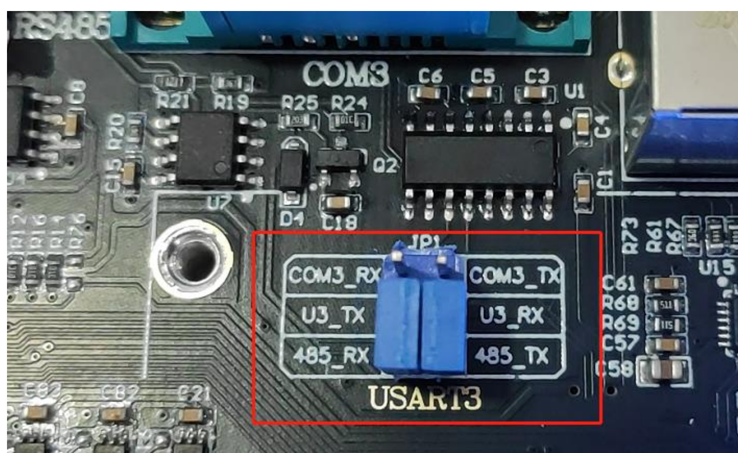


图 3.4.2.1 切换跳线帽的位置

将 USB 转换器上的 RS485 端口的 A 用铜线接到开发板 RA485 端口的 A 处，USB 转换器上的 RS485 端口的 B 用铜线接到开发板 RA485 端口的 B 处。

在计算机的设备查看 USB 转换器的端口号。作者的端口号有两个，一个是开发板 USB 调试串口的，另一个就是 USB 转换器的端口号了。

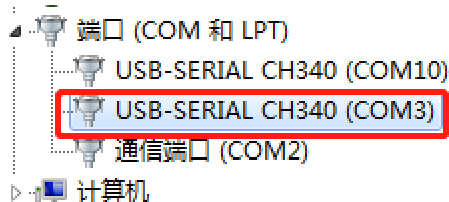


图 3.4.2.2 在设备管理器查看 USB 转换器的端口号

选择正确的 COM 口，波特率为 115200，8N1，无检验位，并建立串口连接。

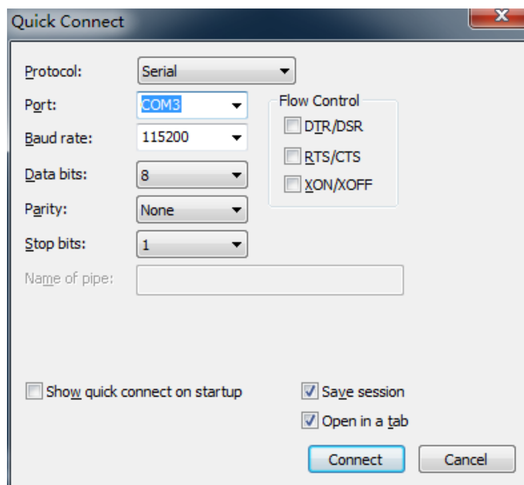


图 3.4.2. 3 串口调试终端设置

下图 serial-com10 为 USB 调试串口的终端，serial-com3 是 usb 转换器的终端。

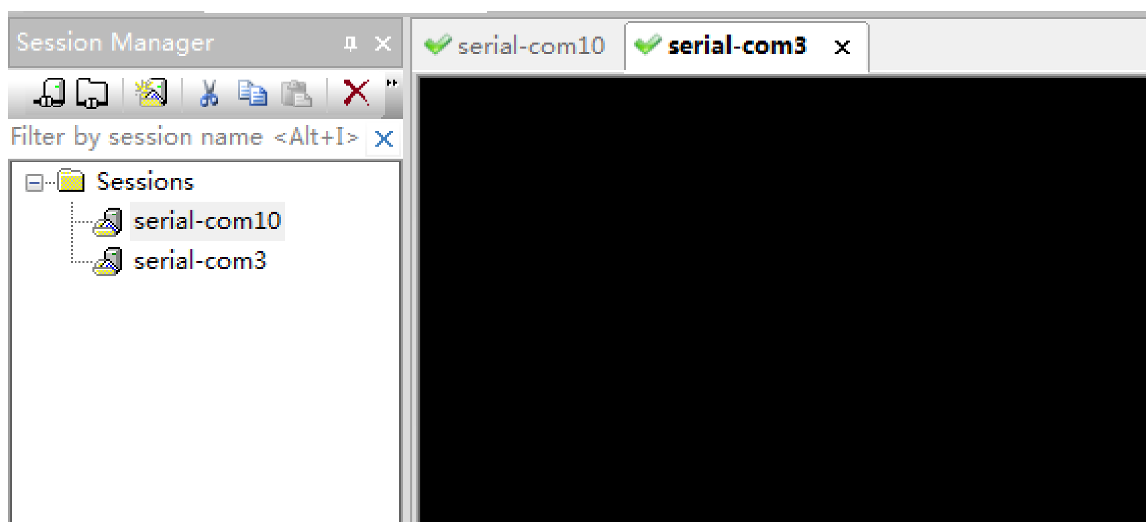


图 3.4.2. 4 连接 COM3

在 usb 串口调试终端输入下面的指令，将 RS485 也设置成一个串口终端。

USER# setsid getty 115200 /dev/ttymxc2

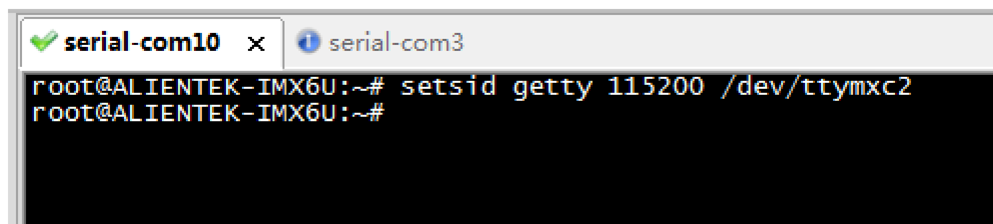


图 3.4.2. 5 使用指令设置 RS485 为一个串口终端

这样可以像调试串口一样输入登录用户名 root，即可进入系统。这样能输入指令并返回结果，表明 RS485 串口正常。

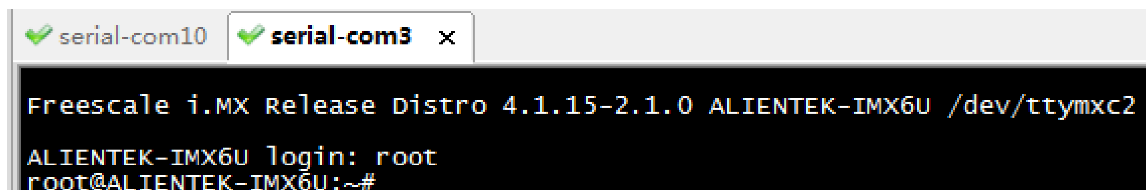


图 3.4.2. 6 输入“root”指令登录开发板

3.5 DDR 测试

Memtester 简单介绍

Memtester 主要是捕获内存错误和一直处于很高或者很低的坏位， 其测试的主要项目有随机值，异或比较，减法，乘法，除法，与或运算等等。 通过给定测试内存的大小和次数，可以对系统现有的内存进行上面项目的测试。

memtester [-p PHYSADDR] <MEMORY> [ITERATIONS]

参数说明：

- MEMORY 申请测试内存的数量，单位默认是 megabytes(兆)，也可以是 B K M G。
- ITERATIONS 测试的次数，默认是无限

使用文件系统自带的 Memtester 测试工具申请 8MB 内存数量测试做 1 次 DDR 测试。执行如下指令。

USER# memtester 8M 1

```
root@ALIENTEK-IMX6:~# memtester 8M 1
memtester version 4.3.0 (32-bit)
Copyright (C) 2001-2012 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffff000
want 8MB (8388608 bytes)
got  8MB (8388608 bytes), trying mlock ...locked.
Loop 1/1:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok

Done.
root@ALIENTEK-IMX6:~#
```

图 3.5 1 测试 DDR

3.6 SD 卡读写测试

本实验测试 SD 卡读写速度，建议使用 SD 卡作为系统启动卡。

指令提示：

time 命令常用于测量一个命令的运行时间，dd 用于复制，从 if(input file)文件读出，写到 of(output file)指定的文件，bs 是每次写块的大小，count 是读写块的数量。"if=/dev/zero"不产生 IO，即可以不断输出数据，因此可以用来测试纯写速度。

3.6.1 SD 卡写速度测试

本次对 SD 系统卡的第一个分区写 50MiB 数据（注意这里读写数据越大测试出来计算出来的结果越平均与接近实际值，但要注意 SD 卡第一个分区的实际大小）。执行下面的指令测试 SD 卡的写速度。

USER# time dd if=/dev/zero of=/run/media/mmcblk0p1/test bs=1024k count=50 conv=fdatasync

```
root@ALIENTEK-IMX6:~# time dd if=/dev/zero of=/run/media/mmcblk0p1/test bs=1024k count=50 conv=fdatasync
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 4.87366 s, 10.8 MB/s

real    0m4.883s
user    0m0.000s
sys     0m2.290s
root@ALIENTEK-IMX6:~#
```

图 3.6.1. 1 执行指令测试 SD 卡的写速度

这里一共写入 50 MiB test 文件，速度为 10.8 MB/s。

3.6.2 SD 卡读速度测试

小提示：

因为 LINUX 的内核机制，一般情况下不需要特意去释放已经使用的 cache。这些 cache 内容可以增加文件以及的读写速度。

执行下面指令清除缓存

USER# echo 3 > /proc/sys/vm/drop_caches

```
root@ALIENTEK-IMX6:~# echo 3 > /proc/sys/vm/drop_caches
[ 1262.284111] sh (639): drop_caches: 3
root@ALIENTEK-IMX6:~#
```

图 3.6.2. 1 执行指令清除缓存

执行下面的指令读取前面用 dd 指令写入的 test 文件

USER# time dd if=/run/media/mmcblk0p1/test of=/dev/null bs=1024k

```
root@ALIENTEK-IMX6:~# time dd if=/run/media/mmcblk0p1/test of=/dev/null bs=1024k
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 2.84218 s, 18.4 MB/s

real    0m2.901s
user    0m0.000s
sys     0m0.480s
root@ALIENTEK-IMX6:~#
```

图 3.6.2. 2 读取 test 文件

这里一共读出了 50 MiB test 文件，速度为 18.4MB/s，测试完成后，可以把 test 文件使用 rm 指令删除。

3.7 NAND FLASH 读写速度测试

本实验测试要求用 SD 卡启动卡启动，使用 NAND FLASH 版本的核心板进行测试。实际上读写数据量越大，数据越平均，越接近实际值。

USER# cat /proc/mtd

```
root@ALIENTEK-IMX6:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00020000 "u-boot"
mtd1: 00020000 00020000 "env"
mtd2: 00100000 00020000 "logo"
mtd3: 00100000 00020000 "dtb"
mtd4: 00800000 00020000 "kernel"
mtd5: 1f1e0000 00020000 "rootfs"
root@ALIENTEK-IMX6:~#
```

图 3.7. 1 查看 Nand Flash 的分区

写数据前需要对操作的分区进行擦除，注意该操作会清除该操作分区的数据，请提前做好数据备份。

USER# flash_erase /dev/mtd5 0 0

```
root@ALIENTEK-IMX6:~# flash_erase /dev/mtd5 0 0
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
root@ALIENTEK-IMX6:~#
```

图 3.7. 2 擦除 mtd5 文件系统分区

执行如下指令往 NAND FLASH 的 mtd5 文件系统分区写入 50MiB 数据，bs 大小为 1024KB

USER# time dd if=/dev/zero of=/dev/mtd5 bs=1024k count=50

```
root@ALIENTEK-IMX6:~# time dd if=/dev/zero of=/dev/mtd5 bs=1024k count=50
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 22.5406 s, 2.3 MB/s

real    0m22.550s
user    0m0.000s
sys     0m9.010s
root@ALIENTEK-IMX6:~#
```

图 3.7. 3 向文件系统分区写入 50MiB 数据

执行如下指令从 NAND FLASH 的 mtd4 内核分区（8MiB）读取 8MiB 数据。

USER# time dd if=/dev/mtd4 of=/dev/null bs=1024k

```
root@ALIENTEK-IMX6:~# time dd if=/dev/mtd4 of=/dev/null bs=1024k
8+0 records in
8+0 records out
8388608 bytes (8.4 MB, 8.0 MiB) copied, 2.43354 s, 3.4 MB/s

real    0m2.443s
user    0m0.000s
sys     0m0.320s
root@ALIENTEK-IMX6:~#
```

图 3.7. 4 读取内核分区数据

3.8 系统时钟与 RTC 时钟

小提示：

测试 RTC 时钟要扣上钮扣电池。

Linux 系统分两个时钟，一个是 system time（软件时钟），一个是 hardware clock（硬件时钟）。使用 date 和 hwclock 命令可分别查看和设定系统时间和硬件时间。系统时钟掉电即会消失，RTC 时钟在有电池的情况下会长期运行。系统时钟会在系统重启时与 RTC 时钟同步。

查看系统时钟，使用指令 date。

USER# date

```
root@ALIENTEK-IMX6:~# date
Tue Jul  9 10:08:47 UTC 2019
root@ALIENTEK-IMX6:~#
```

图 3.8.1 查看系统时钟的时间

查看硬件（RTC）时钟，使用指令 hwclock。

USER# hwclock

```
root@ALIENTEK-IMX6:~# hwclock
Tue Jul  9 10:09:24 2019  0.000000 seconds
root@ALIENTEK-IMX6:~#
```

图 3.8.2 查看硬件时钟

设置系统时钟，设置当前时间，然后查看设置的系统时间。例如当前时间是 2019 年 7 月 9 日 上午 10:00:00。指令如下：

USER# date -s "2019-7-9 10:00:00" // 设置当前系统时钟

USER# date // 查看当前系统时钟

```
root@ALIENTEK-IMX6:~# date -s "2019-7-9 10:00:00"
Tue Jul  9 10:00:00 UTC 2019
root@ALIENTEK-IMX6:~# date
Tue Jul  9 10:00:03 UTC 2019
```

图 3.8.3 设置系统时钟

将系统时钟写入硬件时钟。

USER# hwclock -w // 将系统时钟同步至硬件时钟

USER# hwclock // 查看硬件时钟

```
root@ALIENTEK-IMX6:~# date -s "2019-7-9 10:00:00"
Tue Jul  9 10:00:00 UTC 2019
root@ALIENTEK-IMX6:~# date
Tue Jul  9 10:00:03 UTC 2019
root@ALIENTEK-IMX6:~# hwclock -w
root@ALIENTEK-IMX6:~# hwclock
Tue Jul  9 10:00:08 2019  0.000000 seconds
root@ALIENTEK-IMX6:~#
```

图 3.8.4 将系统时钟写入硬件时钟

3.9 查看系统信息

显示操作系统的内核版本号。

USER# uname -a

```
root@ALIENTEK-IMX6:~# uname -a
Linux ALIENTEK-IMX6 4.1.15+ #7 SMP PREEMPT Wed Jul 10 10:20:01 CST 2019 armv7l armv7l armv7l GNU/Linux
root@ALIENTEK-IMX6:~#
```

图 3.9.1 显示内核版本号

查看系统主机名。

USER# cat /etc/hostname

```
root@ALIENTEK-IMX6:~# cat /etc/hostname
ALIENTEK-IMX6
root@ALIENTEK-IMX6:~#
```

图 3.9.2 查看系统主机名

查看系统登录欢迎信息。

USER# cat /etc/issue

```
root@ALIENTEK-IMX6:~# cat /etc/issue
Freescale i.MX Release Distro 4.1.15-2.1.0 \n \l

root@ALIENTEK-IMX6:~#
```

图 3.9.3 查看系统登录欢迎信息

查看 CPU 相关信息。

USER# cat /proc/cpuinfo

```
root@ALIENTEK-IMX6:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 8.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : Freescale i.MX6 Ultralite (Device Tree)
Revision        : 0000
Serial          : 0000000000000000
root@ALIENTEK-IMX6:~#
```

图 3.9.4 查看 CPU 相关信息

查看内存相关信息。

USER# cat /proc/meminfo

```
root@ALIENTEK:~# cat /proc/meminfo
MemTotal:          507024 kB
MemFree:           358292 kB
MemAvailable:      358144 kB
Buffers:              684 kB
Cached:             69168 kB
SwapCached:             0 kB
Active:             50204 kB
Inactive:           43240 kB
Active(anon):       23292 kB
Inactive(anon):       128 kB
Active(file):       26912 kB
Inactive(file):     43112 kB
Unevictable:            0 kB
Mlocked:                0 kB
HighTotal:              0 kB
HighFree:               0 kB
LowTotal:          507024 kB
LowFree:           358292 kB
SwapTotal:              0 kB
SwapFree:               0 kB
Dirty:                  4 kB
Writeback:              0 kB
AnonPages:          23104 kB
Mapped:             24776 kB
Shmem:                324 kB
Slab:               12004 kB
SReclaimable:        5192 kB
SUnreclaim:          6812 kB
KernelStack:          528 kB
PageTables:           612 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:       253512 kB
Committed_AS:       62312 kB
VmallocTotal:     1548288 kB
VmallocUsed:         3796 kB
VmallocChunk:     1367980 kB
CmaTotal:          131072 kB
CmaFree:            96404 kB
root@ALIENTEK:~#
```

图 3.9. 5 查看内存相关信息

3.10 温度传感器

I.MX6 芯片内部内置有温度传感器，可以实时反映 I.MX6 此 CPU 的内部温度

USER# cat /sys/class/thermal/thermal_zone0/temp

```
root@ALIENTEK-IMX6:~# cat /sys/class/thermal/thermal_zone0/temp
46531
root@ALIENTEK-IMX6:~#
```

图 3.10. 1 查看芯片内部温度

温度值即为 46.531℃（46531/1000）。

3.11 网口测试

小提示：

ALPHA 开发板有 eth0、eth1 两路百兆网卡。eth0 对应底板上 ENET2，eth1 对应底板上的 ENET1。可使用 ifconfig 指令来显示或者配置网络

查看网络信息。

USER# ifconfig

```
root@ALIENTEK-IMX6:~# ifconfig
eth0      Link encap:Ethernet  HWaddr e2:2c:a7:78:59:24
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1548 errors:0 dropped:35 overruns:0 frame:0
          TX packets:76 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:143051 (139.6 KiB)  TX bytes:12362 (12.0 KiB)

eth1      Link encap:Ethernet  HWaddr 2e:e0:f5:6f:06:b9
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:380 (380.0 B)  TX bytes:380 (380.0 B)

root@ALIENTEK-IMX6:~#
```

图 3.11. 1 使用 ifconfig 指令查看网卡信息

插上网线到 ENET2 处可以看到如下信息，系统自动获取了 ip，eth1 同理。

```
root@ALIENTEK-IMX6:~# ifconfig
eth0      Link encap:Ethernet  HWaddr e2:2c:a7:78:59:24
          inet addr:192.168.1.246  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e02c:a7ff:fe78:5924/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1583 errors:0 dropped:35 overruns:0 frame:0
          TX packets:132 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:146425 (142.9 KiB)  TX bytes:21769 (21.2 KiB)

eth1      Link encap:Ethernet  HWaddr 2e:e0:f5:6f:06:b9
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:380 (380.0 B)  TX bytes:380 (380.0 B)

root@ALIENTEK-IMX6:~#
```

图 3.11. 2 查看自动获取的 ip

如果对应网卡没有自动获取到 IP，请使用下面的指令获取。“-i”是指定网卡名称，如不指定，会使用默认会使用 eth0。

USER# udhcpc -i eth0

```
root@ALIENTEK-IMX6:~# udhcpc -i eth0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 192.168.1.247...
Lease of 192.168.1.247 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 114.114.114.114
root@ALIENTEK-IMX6:~#
```

图 3.11.3 使用 udhcpc 手动获取 ip

关闭与打开网口

USER# ifconfig eth1 down　　// 关闭网口，网卡名字请根据实际情况修改，down 表示关闭

USER# ifconfig eth1 up　　// 打开网口，网卡名字请根据实际情况修改，up 表示打开

```
root@ALIENTEK-IMX6:~# ifconfig eth1 down
root@ALIENTEK-IMX6:~# [ 7326.745163] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready

root@ALIENTEK-IMX6:~# ifconfig eth1 up
[ 7335.282788] fec 2188000.ethernet eth1: Freescale FEC PHY driver [Generic PHY] (mii_bus:phy_addr=20b4000.ethernet:00, irq=-1)
root@ALIENTEK-IMX6:~# [ 7337.359690] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[ 7338.283115] fec 2188000.ethernet eth1: Link is Up - 100Mbps/Full - flow control rx/tx
[ 7338.291017] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready

root@ALIENTEK-IMX6:~#
```

图 3.11.4 关闭与打开网口

测试网口是否能上网，以访问 www.baidu.com 为例，执行如下命令，“-I”代表指定网口，不加“-I”则使用默认网卡（默认网卡指的是有网络接入的一端，如果两个网口都有网络接入，则使用 eth0 作为默认网卡）。按“Ctrl+c”终止 ping 指令。百度的实际地址根据网络运营商不同，访问的地址会不同。

USER# ping www.baidu.com -I eth0

```
root@ALIENTEK-IMX6:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.38) from 192.168.1.66 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=56 time=16.5 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=56 time=17.7 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=56 time=10.6 ms
64 bytes from 14.215.177.38: icmp_seq=4 ttl=56 time=5.79 ms
64 bytes from 14.215.177.38: icmp_seq=5 ttl=56 time=6.31 ms
64 bytes from 14.215.177.38: icmp_seq=6 ttl=56 time=9.48 ms
64 bytes from 14.215.177.38: icmp_seq=7 ttl=56 time=7.59 ms
^C64 bytes from 14.215.177.38: icmp_seq=8 ttl=56 time=7.78 ms

--- www.a.shifen.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7010ms
rtt min/avg/max/mdev = 5.791/10.250/17.788/4.274 ms
root@ALIENTEK-IMX6:~#
```

图 3.11.5 测试 eth0 联网

USER# ping www.baidu.com -I eth1

```
root@ALIENTEK-IMX6:~# ping www.baidu.com -I eth1
PING www.a.shifen.com (14.215.177.39) from 192.168.1.249 eth1: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=56 time=11.8 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=56 time=14.7 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=56 time=17.0 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=56 time=10.2 ms
64 bytes from 14.215.177.39: icmp_seq=5 ttl=56 time=14.2 ms
64 bytes from 14.215.177.39: icmp_seq=6 ttl=56 time=13.0 ms
^C
--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 10.208/13.522/17.039/2.183 ms
root@ALIENTEK-IMX6:~#
```

图 3.11.6 测试 eth1 联网

查看网关后，并 ping 网关。

USER# route

```
root@ALIENTEK-IMX6:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    10     0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
root@ALIENTEK-IMX6:~#
```

图 3.11.7 查看网关

由上可知网关为 192.168.1.1，根据路由器不同，网关可能不同。ping 网关可测试内网与开发板连接是否正常。下面指令不加“-I”参数，使用默认网卡。

USER# ping 192.168.1.1

```
root@ALIENTEK-IMX6:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=0.341 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=0.276 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=0.279 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=0.303 ms
^C
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.276/0.299/0.341/0.033 ms
root@ALIENTEK-IMX6:~#
```

图 3.11.8 ping 网关

网络通信速度测试

小提示：

iperf 是一个网络性能测试工具。iperf 可以测试最大 TCP 和 UDP 带宽性能，具有多种参数和 UDP 特性，可以根据需要调整，可以报告带宽、延迟抖动和数据包丢失。

测试 Ubuntu 与开发板通信速度。如果你的 Ubuntu 未安装 iperf，请在 Ubuntu 终端中执行"sudo apt-get install iperf"安装。

查看 Ubuntu 的 ip 地址，备用。（这里要确保开发板的 ip 地址要与 Ubuntu 的 ip 地址是同一局域网内）。

本次测试 Ubuntu 作服务端，开发板作客户端，执行下面指令。

USER# ifconfig

```
alientek@ubuntu:~$ ifconfig
eth0      Link encap:以太网  硬件地址 00:0c:29:c5:42:e8
          inet 地址:192.168.1.84  广播:192.168.1.255  掩码:255.255.255.0
          inet6 地址: fe80::20c:29ff:fec5:42e8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:1999905 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:14047 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:154321542 (154.3 MB)  发送字节:3569406 (3.5 MB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:1100 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:1100 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1
          接收字节:127083 (127.0 KB)  发送字节:127083 (127.0 KB)

alientek@ubuntu:~$
```

图 3.11.9 查看 Ubuntu 的 ip 地址

USER# iperf -s // Ubuntu 作为服务端

```
alientek@ubuntu:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

图 3.11.10 设置 Ubuntu 作为服务端

开发板作为客户端连接 Ubuntu 服务端。

USER# iperf -c 192.168.1.84 -i 1 // -i 1 指通信周期，单位秒。

```
root@ALIENTEK-IMX6:~# iperf -c 192.168.1.84 -i 1
------------------------------------------------------------
Client connecting to 192.168.1.84, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.249 port 54116 connected with 192.168.1.84 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  1.0- 2.0 sec  11.4 MBytes  95.4 Mbits/sec
[  3]  2.0- 3.0 sec  10.8 MBytes  90.2 Mbits/sec
[  3]  3.0- 4.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  4.0- 5.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  5.0- 6.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  6.0- 7.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  7.0- 8.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  8.0- 9.0 sec  11.1 MBytes  93.3 Mbits/sec
[  3]  9.0-10.0 sec  11.2 MBytes  94.4 Mbits/sec
[  3]  0.0-10.0 sec   112 MBytes  93.6 Mbits/sec
root@ALIENTEK-IMX6:~#
```

图 3.11.11 开发板作为客户端并连接 Ubuntu 服务端

Ubuntu 服务端打印如下信息。

```
alientek@ubuntu:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.84 port 5001 connected with 192.168.1.249 port 54116
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec   112 MBytes  93.2 Mbits/sec
```

图 3.11. 12 Ubuntu 服务端打印的信息

反过来 Ubuntu 作客户端，开发板作服务端是一样的结果，这里就不浪费笔墨了。

3.12 FlexCAN 测试

开发板底板有一路 CAN，如果用户手上有测试 CAN 的设备可以参考下面的指令自行测试。本实验使用两块开发板测试，使用它们的 CAN 相互收发数据。

连接方法：将开发板一 CAN 的 H 端子与开发板二 CAN 设备 H 端子连接；开发板一 CAN 的 L 端子与开发板二 CAN 设备 L 端子连接。开发板一作为服务端，开发板二作为客户端。

默认开发板的 can 设备是还没有打开的，使用下面的指令打开 can 设备。

服务端：

USER# ifconfig can0 up

设置 can0 的 can 设备通信波特率为 125000。

USER# ip link set can0 up type can bitrate 125000 triple-sampling on

```
root@ALIENTEK-IMX6U:~# ifconfig can0 up
[  584.030119] flexcan 2090000.can can0: bit-timing not yet defined
SIOCSIFFLAGS: Invalid argument
root@ALIENTEK-IMX6U:~# ip link set can0 up type can bitrate 125000 triple-sampling on
[  599.729606] flexcan 2090000.can can0: writing ctrl=0x0e312085
[  599.737214] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@ALIENTEK-IMX6U:~#
```

3.12. 1 打开 can 设备并设通信波特率

同理，客户端也是这样设置：

USER# ifconfig can0 up

设置 can0 的 can 设备通信波特率为 125000。

USER# ip link set can0 up type can bitrate 125000 triple-sampling on

```
root@ALIENTEK-IMX6U:~# ifconfig can0 up
[  223.772090] flexcan 2090000.can can0: bit-timing not yet defined
SIOCSIFFLAGS: Invalid argument
root@ALIENTEK-IMX6U:~# ip link set can0 up type can bitrate 125000 triple-sampling on
[  230.602294] flexcan 2090000.can can0: writing ctrl=0x0e312085
[  230.610356] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@ALIENTEK-IMX6U:~#
```

3.12. 2 打开 can 设备并设通信波特率

服务端使用 candump 指令接收来自 can0 的数据

USER# candump can0

```
root@ALIENTEK-IMX6U:~# candump can0
```

3.12. 3 服务端等待接收数据

客户端使用 cansend 指令给服务端的 can0 发送数据。指令解释：5A1 为帧 ID，#后面的是数据，共 8 个字节。

USER#　cansend can0 5A1#11.22.33.44.55.66.77.88

```
root@ALIENTEK-IMX6U:~# cansend can0 5A1#11.22.33.44.55.66.77.88
root@ALIENTEK-IMX6U:~#
```

3.12.4 客户端发送数据给服务端

服务端接收到的数据如下：

USER#　candump can0

```
root@ALIENTEK-IMX6U:~# candump can0
  can0  5A1   [8]  11 22 33 44 55 66 77 88
```

3.12.5 服务端接收到的数据

同样地，服务端与客户端的角色互换，执行上面的指令测试即可。

3.13 USB 接口测试

ALPHA 底板 USB/OTG 接口说明：

- ◆ USB_HOST1~USB_HOST3 为 HOST 模式，默认为 HOST 模式；
- ◆ USB1_HOST 支持 HOST 模式；USB_OTG1 支持切换为 HOST/DEVICE

3.13.1 HOST 模式读写测试

以下实验根据使用的 U 盘的不同和实验环境不同，测试结果会有所差异。

将 FAT32 格式 U 盘插到开发板 USB_HOST1~USB_HOST3/USB1_HOST 其中一个接口。插入后会打印如下信息，可以从中看到 U 盘大小和挂载名，如下图所示：

```
root@ALIENTEK-IMX6:~# [  387.252476] usb 1-1.4: new high-speed USB device number 3 using ci_hdrc
[  387.424166] usb-storage 1-1.4:1.0: USB Mass Storage device detected
[  387.435155] scsi host0: usb-storage 1-1.4:1.0
[  388.869693] scsi 0:0:0:0: Direct-Access     Kingston DataTraveler 3.0 PMAP PQ: 0 ANSI: 6
[  388.890489] sd 0:0:0:0: [sda] 30277632 512-byte logical blocks: (15.5 GB/14.4 GiB)
[  388.901085] sd 0:0:0:0: [sda] Write Protect is off
[  388.910109] sd 0:0:0:0: [sda] No Caching mode page found
[  388.916470] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  388.936015]  sda: sda1
[  389.506031] sd 0:0:0:0: [sda] Attached SCSI removable disk
[  390.034185] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.

root@ALIENTEK-IMX6:~#
```

图 3.13.1.1 挂载的 U 盘信息

使用 df -h 查看 U 盘的挂载路径，可以看到下图 U 盘已经挂载在/run/media/下，挂载名为 sda1。

USER#　df -h

```
root@ALIENTEK-IMX6:~# df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root        7.2G  545M  6.3G   8% /
devtmpfs         121M  4.0K  121M   1% /dev
tmpfs             40K     0   40K   0% /mnt/.psplash
tmpfs            121M  160K  121M   1% /run
tmpfs            121M  156K  121M   1% /var/volatile
/dev/mmcblk0p1    63M  6.7M   57M  11% /run/media/mmcblk0p1
/dev/sda1         14G  5.9G  8.2G  42% /run/media/sda1
root@ALIENTEK-IMX6:~#
```

图 3.13.1.2　查看 U 盘的挂载路径

写速度测试：

USER#　time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count=100 conv=fdatasync

```
root@ALIENTEK-IMX6:~# time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count=100 conv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 34.5922 s, 3.0 MB/s

real	0m34.615s
user	0m0.000s
sys	0m3.680s
root@ALIENTEK-IMX6:~#
```

图 3.13.1. 3 写速度测试

本次写 100MiB，速度为 3.0MB/s。

读速度测试：

小提示：

因为 LINUX 的内核机制，一般情况下不需要特意去释放已经使用的 cache。这些 cache 内容可以增加文件以及的读写速度。

执行下面指令清除缓存

USER# echo 3 > /proc/sys/vm/drop_caches

执行下面的指令读取前面用 dd 指令写入的 test 文件。

USER# time dd if=/run/media/sda1/test of=/dev/null bs=1024k

```
root@ALIENTEK-IMX6:~# time dd if=/run/media/sda1/test of=/dev/null bs=1024k
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.61108 s, 22.7 MB/s

real	0m4.621s
user	0m0.000s
sys	0m0.780s
root@ALIENTEK-IMX6:~#
```

图 3.13.1. 4 读速度测试

这里一共读出了 100 MiB test 文件，速度为 22.7 MB/s。

3.13.2 DEVICE 模式测试

本实验将 SD 卡的第一个分区“boot”分区模拟成 U 盘，请将另一根 USB 转串口线接在 USB_OTG1 处，并连接 PC 端。在串口终端执行下面指令，就可以将开发板的 SD 卡模拟成 U 盘挂载在 PC 上。

USER# modprobe g_mass_storage file=/dev/mmcblk0p1 removable=1

```
root@ALIENTEK-IMX6:~# modprobe g_mass_storage file=/dev/mmcblk0p1 removable=1
[ 7189.081096] Mass Storage Function, version: 2009/09/11
[ 7189.086694] LUN: removable file: (no medium)
[ 7189.091274] LUN: removable file: /dev/mmcblk0p1
[ 7189.097729] Number of LUNs=1
[ 7189.100660] Number of LUNs=1
[ 7189.104860] g_mass_storage gadget: Mass Storage Gadget, version: 2009/09/11
[ 7189.111885] g_mass_storage gadget: userspace failed to provide iSerialNumber
[ 7189.119993] g_mass_storage gadget: g_mass_storage ready
root@ALIENTEK-IMX6:~# [ 7189.393008] g_mass_storage gadget: high-speed config #1: Linux File-Backed Storage
```

图 3.13.2. 1 将开发板 boot 分区模拟成一个 U 盘

如下图，已经将 SD 卡的第一个分区成功的挂载在 PC 上。可以当作 U 盘一样使用。

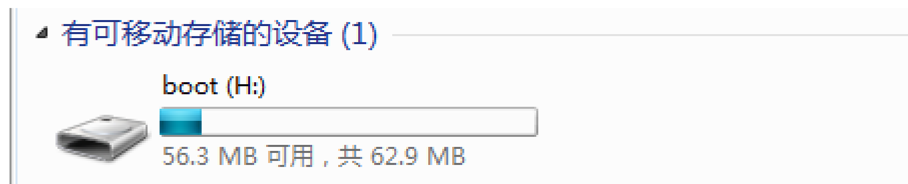


图 3.13.2. 2 模拟成一个 U 盘成功

3.13.3 USB SERIAL 测试

本实验将 USB_OTG1 当作串口使用。将另一根 USB 转串口线接到 USB_OTG1 处，然后连接 PC，在串口终端下执行如下指令。

USER# modprobe g_serial

```
root@ALIENTEK-IMX6:~# modprobe g_serial
[  190.112278] g_serial gadget: Gadget Serial v2.4
[  190.121096] g_serial gadget: g_serial ready
root@ALIENTEK-IMX6:~# [  190.388160] g_serial gadget: high-speed config #2: CDC ACM config

root@ALIENTEK-IMX6:~#
```

图 3.13.3. 1 将 USB_OTG1 模拟成一个串口

查看是否生成/dev/ttyGS0 节点。

USER# ls /dev/ttyGS0

```
root@ALIENTEK-IMX6:~# ls /dev/ttyGS0
/dev/ttyGS0
```

图 3.13.3. 2 查看生成的节点

同时可以在 PC 设备管理器处，查看端口号

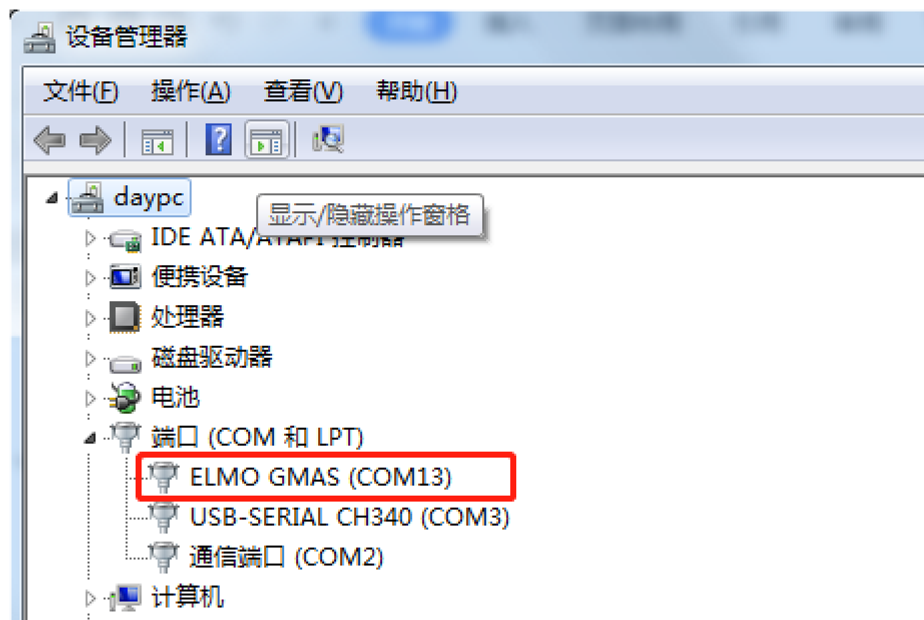


图 3.13.3. 3 在设备管理器查看端口号

开启守护进程

USER# setsid getty 115200 /dev/ttyGS0

```
root@ALIENTEK-IMX6:~# setsid getty 115200 /dev/ttyGS0
root@ALIENTEK-IMX6:~#
```

图 3.13.3. 4 开启守护进程，设置模拟成一个串口终端

然后用 SecureCRT 或者 Putty 软件工具连接到该端口号。打开串口调试终端，选择正确的 COM 口，波特率为 115200，8N1，无检验位，并建立串口连接，按下回车键，可以后其他串口终端一样使用了。

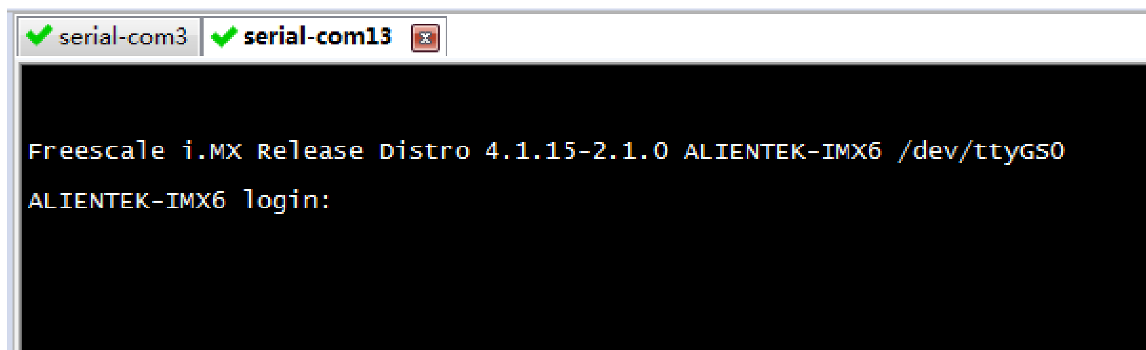


图 3.13.3. 5 在串口终端查看信息

3.14 USB 鼠标测试

说明：使用含 Qt5 的文件系统，启动时插上 RGB 屏幕。

开发板进入系统后，插上鼠标会打印如下信息。

```
root@ALIENTEK-IMX6U:~# [  677.562633] usb 2-1.1: new low-speed USB device number 4 using ci_hdrc
[  677.686761] input: USB OPTICAL MOUSE  as /devices/platform/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb2/2-1/2-1.1/2-1.1:1.0/0003:275D:0BA6.0001/input/input3
[  677.703342] hid-generic 0003:275D:0BA6.0001: input: USB HID v1.11 Mouse [USB OPTICAL MOUSE ] on usb-ci_hdrc.1-1.1/input0

root@ALIENTEK-IMX6U:~#
```

图 3.14. 1 插上鼠标后打印的信息

同时，Qt 桌面也显示了鼠标，可以使用鼠标进行点击操作。

3.15 音频测试

3.15.1 ALSA 简单使用

ALSA（高级 Linux 声音架构）在 Linux 操作系统上提供了音频和 MIDI（Musical Instrument Digital Interface，音乐设备数字化接口）的支持。

amixer 的使用：

USER# amixer --help　　　　// 查看 amixer 的用法说明

```
root@ALIENTEK-IMX6:~# amixer --help
Usage: amixer <options> [command]

Available options:
  -h,--help       this help
  -c,--card N     select the card
  -D,--device N   select the device, default 'default'
  -d,--debug      debug mode
  -n,--nocheck    do not perform range checking
  -v,--version    print version of this program
  -q,--quiet      be quiet
  -i,--inactive   show also inactive controls
  -a,--abstract L select abstraction level (none or basic)
  -s,--stdin      Read and execute commands from stdin sequentially
  -R,--raw-volume Use the raw value (default)
  -M,--mapped-volume Use the mapped volume

Available commands:
  scontrols       show all mixer simple controls
  scontents       show contents of all mixer simple controls (default command)
  sset sID P      set contents for one mixer simple control
  sget sID        get contents for one mixer simple control
  controls        show all controls for given card
  contents        show contents of all controls for given card
  cset cID P      set control contents for one control
  cget cID        get control contents for one control
root@ALIENTEK-IMX6:~#
```

图 3.15.1. 1 查看 amixer 的帮助信息

alsamixer 的使用：

这里不得不说 ALSA 真的是强大啊，amixer 是用指令控制音频设备，alsamixer 则提供一套图形界面来控制音频设备，可以用键盘方向键来控制增减音量，打开或者关闭等。

请注意使用 SecureCRT 或者 putty 时请使用 ssh 登录开发板系统，再使用下面的指令，查看的结果是柱状图，否则在 SecureCRT 直接输入 alsamixer 指令将会看到一些乱码的柱状图，这是因为 AlsaMixer 自带颜色，颜色在串口终端无法完全显示。

USER# alsamixer

图 3.15.1. 2 alsamixer 图形控制界面

3.15.2 Headphone 测试

开发板系统音频输出功能默认是打开的，下面两条指令可不执行。

USER# amixer sset 'Left Output Mixer PCM' on

USER# amixer sset 'Right Output Mixer PCM' on

播放音频文件时插耳机到 PHONE 接口处，可以通过下面的指令来设置耳机的音量，或者通过 alsamixer 图形界面的方法来设置音量。

设置播放音量，执行如下命令，音量的单位是 dB，音量最小为 0，最大为 127。

USER# amixer sset Headphone 110,110 // 耳机音量设置为 52

播放音频

播放开发板文件系统自带的音频，执行下面指令

USER# aplay /usr/share/sounds/alsa/Front_Center.wav

USER# aplay /usr/share/sounds/alsa/Front_Left.wav

USER# aplay /usr/share/sounds/alsa/Front_Right.wav

```
root@ALIENTEK-IMX6:~# aplay /usr/share/sounds/alsa/Front_Center.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Center.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ALIENTEK-IMX6:~# aplay /usr/share/sounds/alsa/Front_Left.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Left.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ALIENTEK-IMX6:~# aplay /usr/share/sounds/alsa/Front_Right.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Right.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ALIENTEK-IMX6:~#
```

图 3.15.2. 1 使用 aplay 播放音频文件

3.15.3 Speaker 测试

ALPHA 开发板底板留出 2 路扬声器接口，一路是接左声道（SPKL），一路是接右声道（SPKR），其中底板的扬声器（小喇叭 8 欧 2W）已经接在右声道（SPKR）上。

ALPHA 开发板底板使用的音频芯片是 WM8960 ，是一款低功耗立体声编解码器，采用 D 类扬声器驱动器，可在 8 欧负载下为每通道提供大于 1 W 功率。

注：测试外接扬声器（喇叭时），不要接插入耳机。

可不执行下面的指令设置扬声器的音量，默认音量为 86。

USER# amixer sset Speaker 110,110 // 扬声器（喇叭）的音量设置为 52

同上面 Headphone 测试一样，执行播放音频的指令测试有声音出来即可。

3.15.4 LINE IN 音频输入测试

为了方便，正点原子提供设置音频输入的脚本和录音脚本，脚本里有注释，脚本内容仅供用户参考。

LINE IN 录音内部无信号放大电路，使用一条 3.5mm 两头均为公头的音频线，一端连接开发板的 LINE IN 接口，另一端连接正在播放音乐的 PC 机或手机。

将 9、测试文件->shell->audio 下的 line_in_config.sh 与 record.sh 拷贝到开发板系统的任意目录下，本次拷贝到/home/root 目录下。若脚本没有可执行权限，请使用 chmod u+x +脚本 指令赋予脚本可执行权限。

执行 line_in_config.sh 脚本配置音频输入模式

USER# ./line_in_config.sh

```
root@ALIENTEK-IMX6U:~# ./line_in_config.sh
numid=1,iface=MIXER,name='Capture Volume'
  ; type=INTEGER,access=rw---R--,values=2,min=0,max=63,step=0
  : values=63,63
  | dBscale-min=-17.25dB,step=0.75dB,mute=0
amixer: Unable to find simple control 'PCM Playback',0

Simple mixer control 'Playback',0
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 255
  Front Left: 255 [100%] [0.00dB]
  Front Right: 255 [100%] [0.00dB]
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Headphone',0
```

图 3.15.4. 1 执行 line_in_config.sh 配置音频输入

执行 record.sh 开始录音（注 record.sh 会录一段 10 秒钟的录音保存为当前目录下名为 record.wav 的音频文件，录制音频完成后会自动播放 record.wav，如果音频播放很小声，请将输入端的音量调高）。

USER# ./record.sh

```
root@ALIENTEK-IMX6U:~# ./record.sh
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [off]
  Front Right: Playback [off]
Simple mixer control 'Headphone',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 0 [0%] [-99999.99dB]
  Front Right: Playback 0 [0%] [-99999.99dB]
Simple mixer control 'Speaker Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [off]
  Front Right: Playback [off]
Simple mixer control 'Speaker',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 0 [0%] [-99999.99dB]
  Front Right: Playback 0 [0%] [-99999.99dB]
Recording WAVE 'record.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Headphone',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 125 [98%] [4.00dB]
  Front Right: Playback 125 [98%] [4.00dB]
Simple mixer control 'Speaker Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Speaker',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 125 [98%] [4.00dB]
  Front Right: Playback 125 [98%] [4.00dB]
Playing WAVE 'record.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
root@ALIENTEK-IMX6U:~#
```

正在制音频文件

播放音频文件

图 3.15.4. 2 执行 record.sh 开始录音，录音完成自动播放

3.15.5 MIC IN 录音测试

为了方便，正点原子提供测试麦克风输入的脚本和录音脚本，脚本里有注释，脚本内容仅供用户参考。

将 9、测试文件->shell->audio 下的 mic_in_config.sh 与 record.sh 拷贝到开发板系统的任意目录下，本次拷贝到/home/root 目录下。若脚本没有可执行权限，请使用 chmod u+x +脚本 指令赋予脚本可执行权限。

USER# ./mic_in_config.sh

```
root@ALIENTEK-IMX6U:~# ./mic_in_config.sh
numid=1,iface=MIXER,name='Capture Volume'
  ; type=INTEGER,access=rw---R--,values=2,min=0,max=63,step=0
  : values=63,63
  | dBscale-min=-17.25dB,step=0.75dB,mute=0
amixer: Unable to find simple control 'PCM Playback',0

Simple mixer control 'Playback',0
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 255
  Front Left: 255 [100%] [0.00dB]
  Front Right: 255 [100%] [0.00dB]
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Headphone',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 125 [98%] [4.00dB]
  Front Right: Playback 125 [98%] [4.00dB]
Simple mixer control 'Speaker Playback ZC',0
```

图 3.15.5. 1 执行 mic_in_config.sh 配置麦克风输入

执行 record.sh 开始录音（注 record.sh 会录一段 10 秒钟的录音保存为当前目录下名为 record.wav 的音频文件，录制音频完成后会自动播放 record.wav，请对准备开发板底板上的麦头进行大声说话进行录音）。

USER# ./record.sh

```
root@ALIENTEK-IMX6U:~# ./record.sh
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [off]
  Front Right: Playback [off]
Simple mixer control 'Headphone',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 0 [0%] [-99999.99dB]
  Front Right: Playback 0 [0%] [-99999.99dB]
Simple mixer control 'Speaker Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [off]
  Front Right: Playback [off]
Simple mixer control 'Speaker',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 0 [0%] [-99999.99dB]
  Front Right: Playback 0 [0%] [-99999.99dB]
Recording WAVE 'record.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo   正在制音频文件
Simple mixer control 'Headphone Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Headphone',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 125 [98%] [4.00dB]
  Front Right: Playback 125 [98%] [4.00dB]
Simple mixer control 'Speaker Playback ZC',0
  Capabilities: pswitch
  Playback channels: Front Left - Front Right
  Mono:
  Front Left: Playback [on]
  Front Right: Playback [on]
Simple mixer control 'Speaker',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 125 [98%] [4.00dB]
  Front Right: Playback 125 [98%] [4.00dB]
Playing WAVE 'record.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo   播放音频文件
root@ALIENTEK-IMX6U:~#
```

图 3.15.5. 2 执行 record.sh 开始录音，录音完成自动播放

3.16 ov5640 摄像头测试

实验前请准备 ov5640 摄像头模块（500 万像素），本公司的任何分辨率的 RCB LCD 电容屏。

摄像头插法：

摄像头镜头往开发板外则直接插到 CAMERA 接口处。由于没有防反插设计，插摄像头时需要注意看底板丝印，按引脚编号对应插上。

本实验使用本公司的 ov5640 模块通过 CSI 总线对视频进行实时采集。

插拔 ov5640 摄像头需要注意：

- 该摄像头不支持热插拔，所以在插拔时都需要断电，再进行操作。
- 插摄像头时请注意底板上的丝印要与摄像头上面的丝印对上，再插上摄像头。

ov5640 在内核里是编译成模块形式的，插上 ov5640 摄像头，板子启动时可以看到如下信息。

```
INIT: version 2.88 booting
Starting udev
[    4.251309] udevd[124]: starting version 3.1.5
[    4.280753] random: udevd urandom read with 76 bits of entropy available
[    5.224646] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[    5.285116] 1-003c supply DOVDD not found, using dummy regulator
[    5.291276] 1-003c supply DVDD not found, using dummy regulator
[    5.385478] 1-003c supply AVDD not found, using dummy regulator
bootlogd: cannot allocate pseudo tty: No such file or directory
[    6.782664] CSI: Registered sensor subdevice: ov5640 1-003c
[    6.788271] camera ov5640, is found
[    8.896399] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
ALSA: Restoring mixer settings...
Wed Jul 10 08:35:50 UTC 2019
```

图 3.16. 1 ov5640 驱动模块加载信息

查看是否生成 video1 节点。（这里可能是 video2，用户根据实际情况查看/dev 下的 video 设备）。

注：一般是 video1 节点就是 ov5640 摄像头的节点，但是也会有特殊情况（需要看 video 设备驱动的加载顺序）。

USER# ls /dev/video1

```
root@ALIENTEK-IMX6:~# ls /dev/video1
/dev/video1
root@ALIENTEK-IMX6:~#
```

图 3.16. 2 查看 ov5640 的节点

查看摄像头支持格式、分辨率及帧率。

USER# v4l2-ctl --device=/dev/video1 --list-formats-ext

```
root@ALIENTEK-IMX6:~# v4l2-ctl --device=/dev/video1 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV-16
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)

root@ALIENTEK-IMX6:~#
```

图 3.16. 3 查看摄像头所支持的格式

本次设置采集的图像分辨率为 1024*768 30fps，如果需要设置其他分辨率采集需要严格按照上面的参数。执行下面指令开始采集，并显示到 LCD 上面，按“Ctrl + c”快捷键终止指令，停止采集。

USER# gst-launch-1.0 -v imxv4l2src device=/dev/video1 ! "video/x-raw, format=(string)YUY2, width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! imxv4l2sink

```
root@ALIENTEK-IMX6U:~# gst-launch-1.0 -v imxv4l2src device=/dev/video1 ! "video/x-raw, format=(string)YUY2,width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! imxv4l2sink
====== IMXV4L2SRC: 4.1.6 build on Aug 14 2019 18:32:57. ======
====== IMXV4L2SINK: 4.1.6 build on Aug 14 2019 18:32:57. ======
Setting pipeline to PAUSED ...
display(/dev/fb0) resolution is (800x480).
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
/GstPipeline:pipeline0/GstImxV4l2Src:imxv4l2src0.GstPad:src: caps = "video/x-raw\,\ format\=\(string\)YUY2\,\ width\=\(int\)1024\,\ height\=\(int\)768\,\ framerate\=\(fraction\)30/1"
/GstPipeline:pipeline0/GstCapsFilter:capsfilter0.GstPad:src: caps = "video/x-raw\,\ format\=\(string\)YUY2\,\ width\=\(int\)1024\,\ height\=\(int\)768\,\ framerate\=\(fraction\)30/1"
/GstPipeline:pipeline0/GstImxV4l2Sink:imxv4l2sink0.GstPad:sink: caps = "video/x-raw\,\ format\=\(string\)YUY2\,\ width[  528.049894] ov5640: actual frame rate of XGA is 22.5fps
\=\(int\)1024\,\ height\=\(int\)768\,\ framerate\=\(fraction\)30/1"
/GstPipeline:pipeline0/GstCapsFilter:capsfilter0.GstPad:sink: caps = "video/x-raw\,\ format\=\(string\)YUY2\,\ width\=\(int\)1024\,\ height\=\(int\)768\,\ framerate\=\(fraction\)30/1"
v4l2sink need allocate 3 buffers.
```

图 3.16.4 执行指令采集图像

屏幕使用本公司 7 寸 800*480 RGB 屏，采集图像十分流畅。图像如下：



图 3.16.5　7 寸屏上的图像

- 保存视频

使用 ov5640 保存视频，本次就不作演示了。指令参考如下，注意对应 ov5640 的节点。录像完成后会在当前目录下保存一个 video.yuv 的视频文件。把它上传到电脑使用 yuv 相关的播放器播放即可。（注意：暂时不支持 rgb 格式采集，因为驱动是 YUYV 格式采集的）

USER# gst-launch-1.0 -vvv -e imxv4l2src num-buffers=1000 device=/dev/video1 ! "video/x-raw, format=(string)YUY2, width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! filesink location=video.yuv

- 拍照功能

使用下面的指令拍照，保存图像为 yuv 格式

USER# gst-launch-1.0 imxv4l2src num-buffers=1 device=/dev/video1 ! 'video/x-raw,format=(string)YUY2,width=1024,height=768' ! filesink location=picture.yuv

使用下面的指令拍照，保存图像为 jpg 格式

USER# gst-launch-1.0 imxv4l2src num-buffers=1 device=/dev/video1 ! jpegenc ! filesink location=picture.jpg

3.17 USB 摄像头测试

实验前请准备 USB 摄像头，符合 UVC（USB video device class）协议的摄像头均可。UVC，全称为：USB video class 或 USB video device class，是 Microsoft 与另外几家设备厂商联合推出的为 USB 视频捕获设备定义的协议标准。符合 UVC 规格的硬件设备在不需要安装任何的驱动程序下即可在主机中正常使用。

插上 USB 摄像头到 USB 接口处，USB 设备驱动打印如下信息

```
root@ALIENTEK-IMX6:~# [   43.872899] usb 2-1.4: new high-speed USB device number 5 using ci_hdrc
[   44.170180] uvcvideo: Found UVC 1.00 device USB Camera (0fde:2024)
[   44.191425] input: USB Camera as /devices/platform/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb2/2-1/2-1.4/2-1.4:1.0/input/input4
[   44.207480] usbcore: registered new interface driver uvcvideo
[   44.215030] USB Video Class driver (1.1.1)
[   44.288964] usbcore: registered new interface driver snd-usb-audio

root@ALIENTEK-IMX6:~#
```

图 3.17. 1 USB 摄像头打印的信息

可以使用指令 lsusb 来查看摄像头信息，可以看到摄像头的 ID 信息。

USER# lsusb

```
root@ALIENTEK-IMX6:~# lsusb
Bus 002 Device 005: ID 0fde:2024
Bus 002 Device 004: ID 0951:1666
Bus 002 Device 003: ID 275d:0ba6
Bus 001 Device 001: ID 1d6b:0002
Bus 002 Device 002: ID 05e3:0608
Bus 002 Device 001: ID 1d6b:0002
root@ALIENTEK-IMX6:~#
```

图 3.17. 2 摄像头的 ID 信息

查看设备节点，video1 是 ov5640 节点，video2 是 USB 摄像头节点。（注这里 video1 不一定是 ov5640 节点，这与 ov5640 驱动和 USB 摄像头它们驱动的加载顺序相关）。

USER# ls /dev/video*

```
root@ALIENTEK-IMX6:~# ls /dev/video*
/dev/video0  /dev/video1  /dev/video2
root@ALIENTEK-IMX6:~#
```

图 3.17. 3 查看 USB 摄像头的节点

查看摄像头支持格式、分辨率及帧率。本次 video1 是 ov5640 设备节点，video2 是 USB 摄像头的节点。

USER# v4l2-ctl --device=/dev/video2 --list-formats-ext

```
root@ALIENTEK-IMX6:~# v4l2-ctl --device=/dev/video2 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'MJPG' (compressed)
        Name        : MJPEG
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 424x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x360
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 960x540
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUV 4:2:2 (YUYV)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 424x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x360
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 960x540
                        Interval: Discrete 0.100s (10.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.100s (10.000 fps)

root@ALIENTEK-IMX6:~#
```

图 3.17.4 查看 USB 摄像头支持的格式

由上图可知看到 USB 摄像头支持两种格式采集，一种是“YUYV”别外一种是“MJPG”。由于使用 gstreamer 使用元件 imxv4l2src 不能设置 MJPG 格式采集，它只能设置 YUYV 格式采集。传入字符参数是 format=(string)YUY2。

本次采集的图像大小为 640*480 30fps，如果需要设置其他分辨率采集需要严格按照上面的参数。执行下面指令开始采集，并显示到 LCD 上面，按“Ctrl + c”终止指令。

USER# gst-launch-1.0 -v imxv4l2src device=/dev/video2 ! "video/x-raw, format=(string)YUY2, width=(int)640, height=(int)480, framerate=(fraction)30/1" ! imxv4l2sink

演示效果图略。

使用 usb 摄像头保存录像，请参考如下指令，注意 usb 摄像头对应的节点名称。录像完成后在当前目录保存了一个 video.mkv 或者 video.mp4 文件，使用 gst-play-1.0/gplay-1.0 直接播放

即可。

USER# gst-launch-1.0 -vvv -e v4l2src num-buffers=1000 device=/dev/video2 ! image/jpeg,width=640,height=480,framerate=30/1,rate=30 ! matroskamux ! filesink location=video.mkv

3.18 EC20 4G 模块上网测试

移远模块自带了一套驱动和拨号软件叫 GobiNet，我们发布的内核源码已经含 GobiNet 驱动。移远 EC20 4G 模块驱动已经编译成模块，可以直接插上 EC20 4G 模块使用。

WWAN LED 指示灯说明，当为低的时候 LED 灯点亮，参考电路如下：

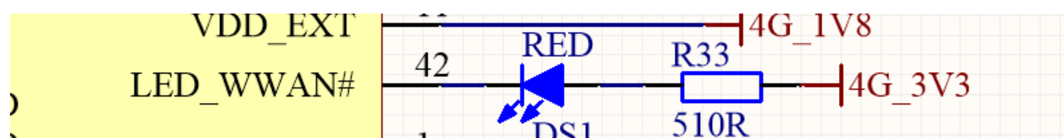


图 3.18. 1 WWWAN LED 指示灯

默认状态下 LED_WWAN 对应的 LED 灯闪烁情况：

引脚工作状态	所指示的网络状态
慢闪(200ms 高/1800ms 低)	找网状态
慢闪(1800ms 高/200ms 低)	待机状态
快闪(125ms 高/125ms 低)	数据传输模式
高电平	通话中

实验前准备：

- EC20 4G 模块
- 4 G 上网卡
- 天线（用于放大信号）

进行 4G 模块测试前，将移动或者联通 4G 卡插到底板的 SIM 卡槽里，再插上 EC20 4G 模块模块，同时插上天线，天线接到模块的 MAIN 处。正确插入 4G 卡与天线后，开发板启动后底板上的 WWAN LED 会亮绿灯，若此灯不亮，请检查 4G 卡是否插对位置，天线是否连接正确。模块安装如下图所示：

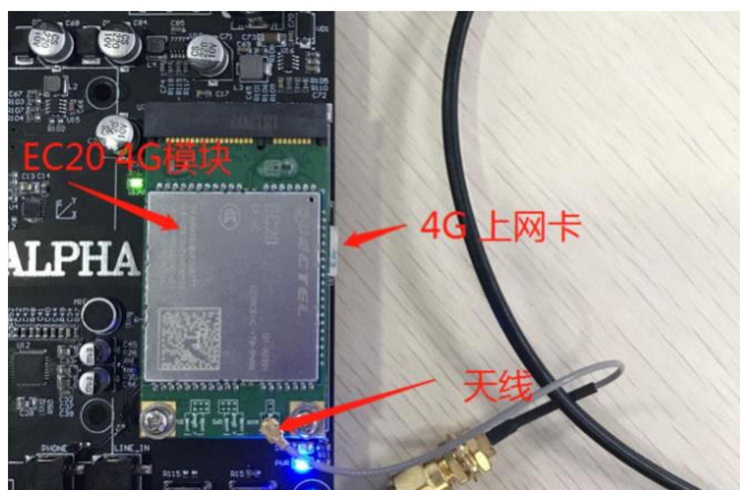


图 3.18. 2 EC20 连接示意图

开发板开机启动打印信息如下：

```
[    5.431700] GobiNet: Quectel_WCDMA&LTE_Linux&Android_GobiNet_Driver_V1.3.0
[    5.470207] GobiNet 2-1.2:1.4 eth2: register 'GobiNet' at usb-ci_hdrc.1-1.2, GobiNet Ethernet Device, 02:29:21:62:25:3e
[    5.551631] creating qcqmi2
[    5.565414] usbcore: registered new interface driver GobiNet
[    5.625455] 1-003c supply DOVDD not found, using dummy regulator
[    5.631608] 1-003c supply DVDD not found, using dummy regulator
[    5.639133] usbcore: registered new interface driver usbserial
[    5.688134] usbcore: registered new interface driver usbserial_generic
[    5.712643] 1-003c supply AVDD not found, using dummy regulator
[    5.723145] usbserial: USB Serial support registered for generic
[    5.767634] usbcore: registered new interface driver option
[    5.829373] usbserial: USB Serial support registered for GSM modem (1-port)
[    5.862587] ov5640_read_reg:write reg error:reg=300a
[    5.870597] option 2-1.2:1.0: GSM modem (1-port) converter detected
[    5.888128] camera ov5640 is not found
[    5.923231] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB0
[    5.964444] option 2-1.2:1.1: GSM modem (1-port) converter detected
[    6.008345] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB1
[    6.065691] option 2-1.2:1.2: GSM modem (1-port) converter detected
[    6.112428] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB2
[    6.159666] option 2-1.2:1.3: GSM modem (1-port) converter detected
[    6.196485] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB3
[    6.277137] random: nonblocking pool is initialized
[    8.726086] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[    9.645920] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[   10.013908] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
bootlogd: cannot allocate pseudo tty: No such file or directory
Populating dev cache
tar: dev/disk/by-label/xc0xcf\303\253xccxd2Uxc5xcc: Cannot stat: No such file or directory
tar: Exiting with failure status due to previous errors
udev-cache: update failed!
ALSA: Restoring mixer settings...
Found hardware: "wm8960-audio" "" "" "" ""
Hardware is initialized using a generic method
/usr/sbin/alsactl: set_control:1325: failed to obtain info for control #58 (No such file or directory)
INIT: Entering runlevel: 5
```

图 3.18. 3 EC20 打印的 USB 信息

查看是否存在/dev/qcqmil2 节点，如果存在的话就说明 GobiNet 驱动成功，如下图所示：

USER# ls /dev/q*

```
root@ALIENTEK-IMX6:~# ls /dev/q*
/dev/qcqmi2
root@ALIENTEK-IMX6:~#
```

图 3.18. 4 查看 EC20 生成的节点

再查看是否生成/dev/ttyUSB0~3 节点

USER# ls /dev/ttyUSB*

```
root@ALIENTEK-IMX6:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3
root@ALIENTEK-IMX6:~#
```

图 3.18. 5 生成的 USB 节点

这四路 ttyUSB 的功能如下图图 3.18. 6 所示，不测试全部测试这些功能了，这里我们只测试上网功能。详细请自行参考 EC20 4G 模块手册。

Product			USB Driver	Interface
UC15	VID: 0x05c6	PID: 0x9090	USB Serial	ttyUSB0 → DM
UC20	VID: 0x05c6	PID: 0x9003		ttyUSB1 → For GPS NMEA message output
EC25	VID: 0x2c7c	PID: 0x0125		ttyUSB2 → For AT commands
EC21	VID: 0x2c7c	PID: 0x0121		ttyUSB3 → For PPP connections or AT commands
EC20	VID: 0x05c6	PID: 0x9215		
EC20 R2.0	VID: 0x2c7c	PID: 0x0125		
EG91	VID: 0x2c7c	PID: 0x0191		
EG95	VID: 0x2c7c	PID: 0x0195		
UC20	VID: 0x05c6	PID: 0x9003	GobiNet or QMI WWAN	ethX or wwanX → Interface 4 can be used as USB Network Adapter
EC25	VID: 0x2c7c	PID: 0x0125		
EC21	VID: 0x2c7c	PID: 0x0121		
EC20	VID: 0x05c6	PID: 0x9215		
EC20 R2.0	VID: 0x2c7c	PID: 0x0125		
EG91	VID: 0x2c7c	PID: 0x0191		
EG95	VID: 0x2c7c	PID: 0x0195		
EG06	VID: 0x2c7c	PID: 0x0306		
EP06	VID: 0x2c7c	PID: 0x0306		
EM06	VID: 0x2c7c	PID: 0x0306		
BG96	VID: 0x2c7c	PID: 0x0296		

图 3.18.6 四路 ttyUSB 的功能示意图

图 3.18.7

3.18.1 拨号上网

使用 quectel-CM 拨号程序工具（这个工具是我们预先交叉编译好放进文件系统/usr/sbin 目录下面的），方便用户使用。

执行下面的指令，等待指令执行完成，如下图所示：

USER# quectel-CM -s cenet &

```
root@ALIENTEK-IMX6:~# quectel-CM -s cenet &
[1] 679
root@ALIENTEK-IMX6:~# [06-04_11:06:43:392] WCDMA&LTE_QConnectManager_Linux&Android_V1.1.34
[06-04_11:06:43:394] quectel-CM profile[1] = cenet///0, pincode = (null)
[06-04_11:06:43:399] Find /sys/bus/usb/devices/2-1.2 idVendor=2c7c idProduct=0125
[06-04_11:06:43:399] Find /sys/bus/usb/devices/2-1.2:1.4/net/eth2
[06-04_11:06:43:400] Find usbnet_adapter = eth2
[06-04_11:06:43:401] Find /sys/bus/usb/devices/2-1.2:1.4/GobiQMI/qcqmi2
[06-04_11:06:43:401] Find qmichannel = /dev/qcqmi2
[06-04_11:06:43:442] Get clientWDS = 7
[06-04_11:06:43:474] Get clientDMS = 8
[06-04_11:06:43:506] Get clientNAS = 9
[06-04_11:06:43:538] Get clientUIM = 10
[06-04_11:06:43:570] Get clientWDA = 11
[06-04_11:06:43:602] requestBaseBandVersion EC20CEHCR06A03M1G
[06-04_11:06:43:698] requestGetSIMStatus SIMStatus: SIM_READY
[06-04_11:06:43:699] requestSetProfile[1] cenet///0
[06-04_11:06:43:762] requestGetProfile[1] cenet///0
[06-04_11:06:43:794] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[06-04_11:06:43:826] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[06-04_11:06:43:890] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[06-04_11:06:43:921] requestSetupDataCall WdsConnectionIPv4Handle: 0xe1811930
[   24.151594] IPv6: ADDRCONF(NETDEV_CHANGE): eth2: link becomes ready
[06-04_11:06:44:018] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[06-04_11:06:44:050] ifconfig eth2 up
[06-04_11:06:44:084] busybox udhcpc -f -n -q -t 5 -i eth2
[06-04_11:06:44:110] udhcpc (v1.24.1) started
[   24.423126] GobiNet 2-1.2:1.4 eth2: kevent 12 may have been dropped
[06-04_11:06:44:307] Sending discover...
[06-04_11:06:44:367] Sending select for 10.6.143.117...
[06-04_11:06:44:427] Lease of 10.6.143.117 obtained, lease time 7200
[06-04_11:06:44:566] /etc/udhcpc.d/50default: Adding DNS 120.196.165.7
[06-04_11:06:44:571] /etc/udhcpc.d/50default: Adding DNS 221.179.38.7

root@ALIENTEK-IMX6:~#
```

图 3.18.1. 1 执行指令拨号上网

查看网卡信息，使用 ifconfig 指令，其中 eth2 就是 4G 网卡。可以看到 quectel-CM 还获取了 ip 10.6.143.117。

```
root@ALIENTEK-IMX6:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 3a:0a:6a:f2:04:91
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 2e:87:5e:78:cf:59
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth2      Link encap:Ethernet  HWaddr f2:ac:d7:fd:80:28
          inet addr:10.6.143.117  Bcast:10.6.143.119  Mask:255.255.255.252
          UP BROADCAST RUNNING NOARP MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2074 (2.0 KiB)  TX bytes:5780 (5.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:380 (380.0 B)  TX bytes:380 (380.0 B)

root@ALIENTEK-IMX6:~#
```

图 3.18.1. 2 查看网卡信息

3.18.2 上网测试

为了准确测试，测试前请把其他网口网线拔出或者关掉。使用 ifconfig 指令关掉其他网卡。

```
root@ALIENTEK-IMX6U:~# ifconfig eth0 down
root@ALIENTEK-IMX6U:~# ifconfig eth1 down
root@ALIENTEK-IMX6U:~#
```

图 3.18.2. 1 关闭其他网卡

执行下面的指令测试上网，参数“-I”指定网卡。这里要指定 eth2 网卡。可按“Ctrl + c”终止 ping 指令。出现如下结果，说明 ping 百度成功。

USER# ping www.baidu.com -I eth2

```
root@ALIENTEK-IMX6:~# ping www.baidu.com -I eth2
PING www.a.shifen.com (183.232.231.174) from 10.6.143.117 eth2: 56(84) bytes of data.
64 bytes from 183.232.231.174: icmp_seq=1 ttl=56 time=50.1 ms
64 bytes from 183.232.231.174: icmp_seq=2 ttl=56 time=33.4 ms
64 bytes from 183.232.231.174: icmp_seq=3 ttl=56 time=24.2 ms
64 bytes from 183.232.231.174: icmp_seq=4 ttl=56 time=31.1 ms
64 bytes from 183.232.231.174: icmp_seq=5 ttl=56 time=44.2 ms
^C64 bytes from 183.232.231.174: icmp_seq=6 ttl=56 time=28.5 ms

--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 24.249/35.318/50.151/9.023 ms
root@ALIENTEK-IMX6:~#
```

图 3.18.2. 2 执行指令 ping 百度

3.19 视频播放测试

文件系统含 GStreamer（流媒体应用的开源多媒体框架），它采用基于插件（plugin）和管道（pipeline）体系结构，提供了 GStreamer 相关的库，同时提供了相关的应用程序。简单来说，用户可以把它当作一种多媒体播放器。然后我们就可以使用播放器播放我们的音频和视频了，但 GStreamer 的功能远远不仅仅是这些。

由于 CPU 没有硬件多媒体解码器且 CPU 资源有限，播放视频分辨率不能太高，帧率也不要太高，否则播放会有卡顿感。

播放视频测试，执行下面指令，播放系统目录/opt/meida/test_movie.avi，分辨率为 856x480，25 帧的视频文件。

USER# gst-play-1.0 /opt/media/test_movie.avi

```
root@ALIENTEK-IMX6:~# gst-play-1.0 /opt/meida/test_movie.avi
Press 'k' to see a list of keyboard shortcuts.
Now playing /opt/meida/test_movie.avi
Prerolling...
====== AIUR: 4.1.6 build on Jun  4 2019 17:31:23. ======
        Core: AVI_PARSER_03.05.28  build on Oct 10 2016 07:21:58
 file: /usr/lib/imx-mm/parser/lib_avi_parser_arm11_elinux.so.3.1
------------------------
    Track 00 [video_0] Enabled
        Duration: 0:00:20.040000000
        Language: und
    Mime:
        video/mpeg, systemstream=(boolean)false, parsed=(boolean)true, mpegversion=(int)4, width=(int)856, height=(int)480, framerate=(fraction)25/1
------------------------
====== IMXV4L2SINK: 4.1.6 build on Jun  4 2019 17:31:35. ======
display(/dev/fb0) resolution is (1280x800).
------------------------
    Track 01 [audio_0] Enabled
        Duration: 0:00:20.062040000
        Language: und
    Mime:
        audio/mpeg, mpegversion=(int)1, channels=(int)2, rate=(int)44100, bitrate=(int)0
------------------------
Redistribute latency...
Home directory not accessible: Permission denied
Home directory not accessible: Permission denied

====== BEEP: 4.1.6 build on Jun  4 2019 17:31:27. ======
        Core: MP3 decoder Wrapper  build on Mar 21 2014 15:04:50
 file: /usr/lib/imx-mm/audio-codec/wrap/lib_mp3d_wrap_arm12_elinux.so.3
CODEC: BLN_MAD-MMCODECS_MP3D_ARM_02.13.00_CORTEX-A8  build on Jul 12 2016 13:15:30.
Redistribute latency...
```

图 3.19. 1 执行指令播放视频

LCD 屏幕上播放的视频：



图 3.19. 2 LCD 屏幕上播放的视频

3.20 AP3612C 测试

AP3612C 简介：

ALPHA 开发板上通过 I2C1 连接了一个三合一环境传感器：AP3216C，AP3216C 是由敦南可以推出的一款传感器，其支持环境光强度(ALS)、接近距离(PS)和红外线强度(IR)这三个环境参数检测。AP3216C 的特点如下：

①、I2C 接口，快速模式下波特率可以到 400Kbit/S

②、多种工作模式选择：ALS、PS+IR、ALS+PS+IR、PD 等等。

③、内建温度补偿电路。

④、宽工作温度范围(-30°C ~ +80°C)。

⑤、超小封装，4.1mm x 2.4mm x 1.35mm

⑥、环境光传感器具有 16 为分辨率。

⑦、接近传感器和红外传感器具有 10 为分辨率。

AP3216C 常被用于手机、平板、导航设备等，其内置的接近传感器可以用于检测是否有物体接近，比如手机上用来检测耳朵是否接触听筒，如果检测到的话就表示正在打电话，手机就会关闭手机屏幕以省电。也可以使用环境光传感器检测光照强度，可以实现自动背光亮度调节。

进入开发板文件系统执行下面指令读取环境传感器的环境参数值，根据开发板所处环境不同，环境参数值不同，先用下面指令读取一次环境参数值，再用手接近 AP3216C 传感器（ALPHA 底板 U8 处），再用指令读取相应的参数值，参数值会有比较大的变化。

读取环境光强度值（ALS）

USER# cat /sys/class/misc/ap3216c/als

```
root@ALIENTEK-IMX6:~# cat /sys/class/misc/ap3216c/als
33
root@ALIENTEK-IMX6:~#
```

图 3.20. 1 读取环境光强度值

读取接近距离(PS)

USER# cat /sys/class/misc/ap3216c/ps

```
root@ALIENTEK-IMX6:~# cat /sys/class/misc/ap3216c/ps
23
root@ALIENTEK-IMX6:~#
```

图 3.20. 2 读取接近距离值

USER# cat /sys/class/misc/ap3216c/ir

读取红外线强度(IR)

```
root@ALIENTEK-IMX6:~# cat /sys/class/misc/ap3216c/ir
16
root@ALIENTEK-IMX6:~#
```

图 3.20. 3 读取红外线强度值

3.21 icm20608 测试

ICM-20608 简介：

ICM-20608 是 InvenSense 出品的一款 6 轴 MEMS 传感器，包括 3 轴加速度和 3 轴陀螺仪。

ICM-20608 尺寸非常小，只有 3x3x0.75mm，采用 16P 的 LGA 封装。ICM-20608 内部有一个 512 字节的 FIFO。陀螺仪的量程范围可以编程设置，可选择±250，±500，±1000 和±2000°/s，加速度的量程范围也可以编程设置，可选择±2g，±4g，±4g，±8g 和±16g。陀螺仪和加速度计都是 16 位的 ADC，并且支持 I2C 和 SPI 两种协议，使用 I2C 接口的话通信速度最高可以达到 400KHz，使用 SPI 接口的话通信速度最高可达到 8MHz。I.MX6U-ALPHA 开发板上的 ICM-20608 通过 SPI 接口和 I.MX6U 连接在一起。ICM-20608 特性如下：

①、陀螺仪支持 X，Y 和 Z 三轴输出，内部集成 16 位 ADC，测量范围可设置：±250，±500，±1000 和±2000°/s。

②、加速度计支持 X，Y 和 Z 轴输出，内部集成 16 位 ADC，测量范围可设置：±2g，±4g，±4g，±8g 和±16g。

③、用户可编程中断。

④、内部包含 512 字节的 FIFO。

⑤、内部包含一个数字温度传感器。

⑥、耐 10000g 的冲击。

⑦、支持快速 I2C，速度可达 400KHz。

⑧、支持 SPI，速度可达 8MHz。

I.MX6U-ALPHA 使用 SPI3 接口连接了一个六轴传感器 ICM-20608，由正点原子提供 linux 驱动程序与用户测试程序。

将 9、测试文件->driver->icm20608 目录下的 icm20608.ko 驱动文件与用户测试程序 icm20608App 拷贝到开发板的任意目录下，本次拷贝到/home/root 目录下（开发板默认已经拷贝到时/home/root 目录下），如下图所示

```
root@ALIENTEK-IMX6U:~# ls
icm20608.ko  icm20608App
root@ALIENTEK-IMX6U:~#
```

图 3.21.1 拷贝驱动文件与程序程序到文件系统

使用 insmod 指令安装驱动文件 icm20608.ko，安装成功如下图。驱动在安装时会读取 ICM-20608 的 ID 信息，读取返回 0XAE 表明驱动正常。

USER# insmod icm20608.ko

```
root@ALIENTEK-IMX6U:~# insmod icm20608.ko
[ 1934.583835] ICM20608 ID = 0XAE
root@ALIENTEK-IMX6U:~#
```

图 3.21.2 安装驱动文件

同时在/dev/生成 icm20608 节点。

USER# ls /dev/icm20608

```
root@ALIENTEK-IMX6U:~# ls /dev/icm20608
/dev/icm20608
root@ALIENTEK-IMX6U:~#
```

图 3.21.3 查看/dev 下生成的 icm20608 节点

执行下面的指令可获取 6 轴传感器的值，因为用户程序 icm20608App 有中文信息打印，所以用户的 securcrt 串口调试终端需要设置面 UTF-8 编码以防止打印时有中文乱码。

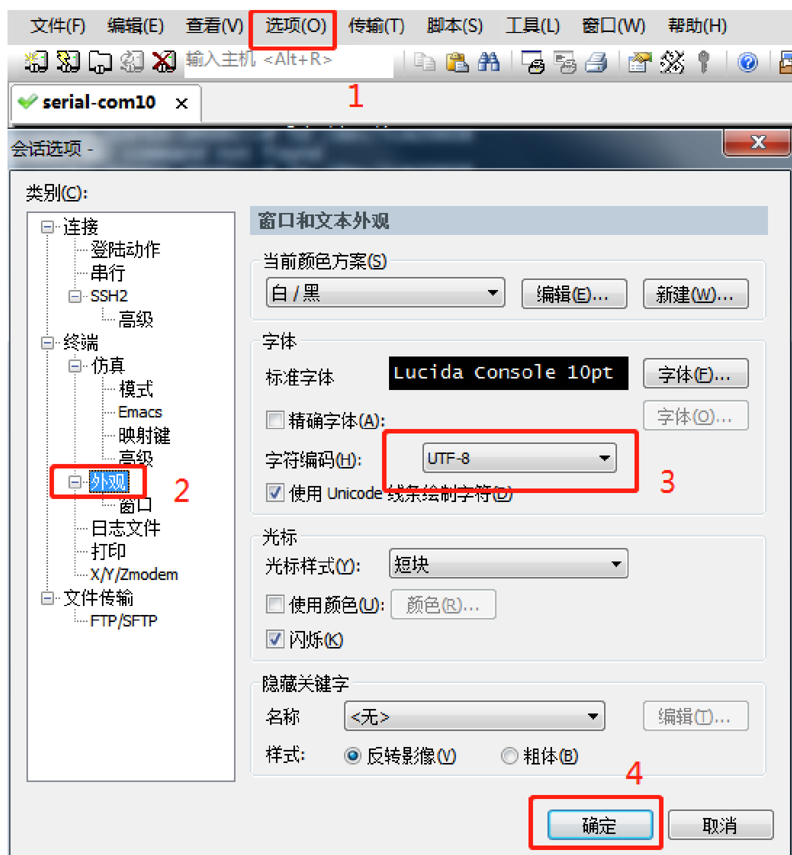


图 3.21.4 串口调试终端调置 UTF-8 编码

在当前目录下输入下面的指令获取 ICM-20608 内部数据。

USER# ./icm20608App /dev/icm20608

```
root@ALIENTEK-IMX6U:~# ./icm20608App /dev/icm20608

原始值:
gx = -9, gy = 10, gz = 2
ax = -136, ay = 5, az = 2088
temp = 2495
实际值:act gx = -0.55°/S, act gy = 0.61°/S, act gz = 0.12°/S
act ax = -0.07g, act ay = 0.00g, act az = 1.02g
act temp = 32.56°C

原始值:
gx = -10, gy = 11, gz = 1
ax = -137, ay = 7, az = 2087
temp = 2488
实际值:act gx = -0.61°/S, act gy = 0.67°/S, act gz = 0.06°/S
act ax = -0.07g, act ay = 0.00g, act az = 1.02g
act temp = 32.54°C

原始值:
gx = -11, gy = 11, gz = 1
ax = -135, ay = 5, az = 2079
temp = 2491
实际值:act gx = -0.67°/S, act gy = 0.67°/S, act gz = 0.06°/S
act ax = -0.07g, act ay = 0.00g, act az = 1.02g
act temp = 32.55°C
```

图 3.21.5 执行指令获取内部数据

3.22 USB WIFI 模块测试

本实验使用 USB WIFI RTL8188EUS 模块（可直接插到电脑上联网测试好坏），传输速率为 150Mbps，使用 USB 2.0 HOST 接口，ALPHA 底板 4 个 USB 接口都可以。正点原子提

供 USB WIFI 测试脚本 alientek_usb_wifi_setup.sh，仅供用户参考。

测试前准备 USB WIFI RTL8188EUS 模块，一般 USB 设备都是可带电插拔的，同理我们的 RTL8188EUS 模块也是支持热插拔的，可在系统起来后再插上 USB WIFI 模块。

拷贝 9、测试文件->shell->wifi->alientek_usb_wifi_setup.sh 到开发板的任意路径下。本次拷贝到/home/root 目录下（开发板默认已经拷贝到时/home/root 目录下））， 如下图

USER# ls

```
root@ALIENTEK-IMX6U:~# ls
alientek_usb_wifi_setup.sh
root@ALIENTEK-IMX6U:~#
```

图 3.22.1 拷贝 WIFI 测试脚本到文件系统目录

开发板在 USB 接口处插上 USB WIFI RTL8188EUS 模块，如下图（下图为 ALPHA 底板）。

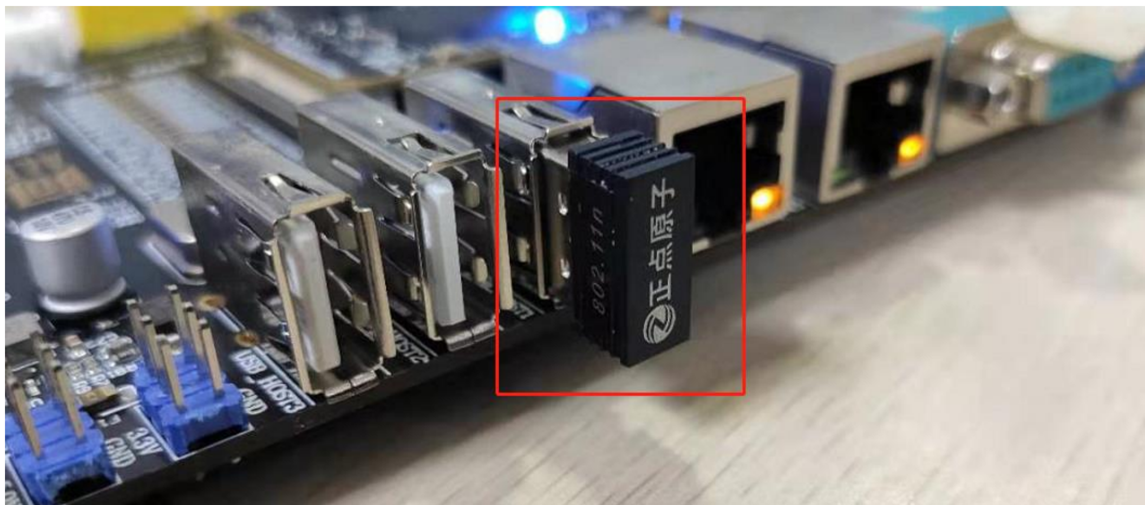


图 3.22.2 USB WIF 连接示意图

开机后插上 USB WIFI，驱动打印信息如下：

```
root@ALIENTEK:~# [   33.112430] usb 2-1.1: new full-speed USB device number 3 using ci_hdrc
[   33.272420] usb 2-1.1: new high-speed USB device number 4 using ci_hdrc
[   33.479535] RTL871X: module init start
[   33.486862] RTL871X: rtl8188eu v4.3.0.9_15178.20150907
[   33.498061] bFWReady == _FALSE call reset 8051...
[   33.547879] RTL871X: rtw_ndev_init(wlan0)
[   33.566133] usbcore: registered new interface driver rtl8188eu
[   33.572054] RTL871X: module init ret=0
[   34.183335] ==> rtl8188e_iol_efuse_patch
[   34.515256] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[   34.523644] RTL871X: set bssid:00:00:00:00:00:00
[   34.529069] RTL871X: set ssid [g_F|Z.c3^g fw_state=0x00000008
[   35.849620] RTL871X: indicate disassoc
```

图 3.22.3 打印的驱动信息

查看 USB WIFI 的网卡信息，使用 ifconfig 指令，如下图示，wlan0 是 USB WIFI 的节点。

USER# ifconfig

```
root@ALIENTEK:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b6:76:0f:ac:f5:15
          inet addr:192.168.1.67  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::b476:fff:feac:f515/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5206 errors:0 dropped:12 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:402447 (393.0 KiB)  TX bytes:12094 (11.8 KiB)

eth1      Link encap:Ethernet  HWaddr 5a:3a:65:65:ea:ba
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1509 (1.4 KiB)  TX bytes:1509 (1.4 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:13:ef:80:38:f7
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:10 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ALIENTEK:~#
```

图 3.22.4 查看网卡信息

3.22.1 Station（上网）模式

本次实验目的：使用 USB WIFI 连接无线网络并测试网络是否能上网。

扫描附近无线网络信息并打印，如下图扫描到本公司的无线网络名称“ALIENTEK”，还能看到加密类型等。（备注：还可以用其他指令来扫描无线网络信息，如 iwlist wlan0 scan）

USER# wpa_cli -i wlan0 scan_result

```
root@ALIENTEK:~# wpa_cli -i wlan0 scan_result
bssid / frequency / signal level / flags / ssid
e4:0e:ee:f2:11:14       2412    64      [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][WPS][ESS]       ALIENTEK
88:f8:72:8d:f3:18       2412    100     [WPA2-PSK-CCMP][WPS][ESS]       ZZK
88:f8:72:8d:f3:19       2412    100     [WPA2-PSK-CCMP][ESS]
00:6b:8e:fb:af:50       2412    100     [WPA2-PSK-CCMP][ESS]    PDCN
48:ee:0c:44:c7:ca       2412    92      [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Stone123456
e4:0e:ee:f2:11:19       2412    60      [WPA2-PSK-CCMP][ESS]
bc:5f:f6:a5:f5:9f       2472    58      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      FAE
0c:4b:54:1f:04:1a       2462    23      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      708
e6:ea:83:c0:a8:f5       2437    23      [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    hnc2016
e0:1d:3b:04:d5:04       2452    23      [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] ChinaNet-cYXL
02:34:cb:b8:46:d0       2412    44      [ESS]   Wireless_BY
root@ALIENTEK:~#
```

图 3.22.1.1 扫描附近的无线网络信息

USER# source ./alientek_usb_wifi_setup.sh -m station -i ALIENTEK -p 15902020353 -d wlan0

参数解释：

-m station ： 设置成 station 模式

-i ALIENTEK ： 无线网络名称(ssid)。

-p 15902020353 ： 无线网络密码(psk)。

-d wlan0 ： USB WIFI 节点

看到下图已经获取到 ip 信息就代表连接无线网络成功，如果没有获取到 ip 信息，请检查密码是否正确或者重试指令。

```
root@ALIENTEK:~# source ./alientek_usb_wifi_setup.sh -m station -i ALIENTEK -p 15902020353 -d wlan0
您的WIFI配置信息是:
mode   : station
ssid   : ALIENTEK
psk    : 15902020353
device: wlan0
waln0: ERROR while getting interface flags: No such device
QObject::disconnect: Unexpected null parameter
[  109.933181] ==> rtl8188e_iol_efuse_patch
[  110.259909] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[  110.267028] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
Successfully initialized wpa_supplicant
[  112.848367] RTL871X: set bssid:00:00:00:00:00:00
ioctl[SIOCSIWAP]: Operation not p[  112.853867] RTL871X: set ssid [g F|Z.c3w_state=0x00000008
ermitted
udhcpc (v1.24.1) started
Sending discover...
[  114.179446] RTL871X: indicate disassoc
[  114.193962] RTL871X: set ssid [ALIENTEK] fw_state=0x00000008
[  114.200870] RTL871X: set bssid:e4:0e:ee:f2:11:14
[  114.257888] RTL871X: start auth
[  114.263368] RTL871X: auth success, start assoc
[  114.273932] RTL871X: assoc success
[  114.277765] RTL871X: recv eapol packet
[  114.281615] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[  114.300396] RTL871X: send eapol packet
[  114.317540] RsvdPageNum: 8
[  115.271720] RTL871X: recv eapol packet
[  115.276269] RTL871X: send eapol packet
[  115.292614] RTL871X: recv eapol packet
[  115.298308] RTL871X: send eapol packet
[  115.306000] RTL871X: set pairwise key camid:4, addr:e4:0e:ee:f2:11:14, kid:0, type:AES
[  115.320556] RTL871X: set group key camid:5, addr:e4:0e:ee:f2:11:14, kid:1, type:TKIP
Sending discover...
Sending select for 192.168.2.211...
Lease of 192.168.2.211 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.2.1
SIOCADDRT: Network is unreachable
WIFI设置station模式完成!
root@ALIENTEK:~#
```

图 3.22.1. 2 执行指令连接无线网络

(注：其中打印的 waln0: ERROR while getting interface flags: No such device 错误信息可以忽略。)

若未能正常获取 ip，等待 RTL871X: set group key camid:5 这句话出现后输入 udhcpc -i wlan0 指令重新获取 ip。若没有这句话，请检查无线网络信息是否正确。

测试是否能上网，使用 ping 指令 ping 百度，可按 ctrl+c 终止执行指令

USER# ping www.baidu.com

看到如下信息，就代表能上网

```
root@ALIENTEK:~# ping www.baidu.com
PING www.a.shifen.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=55 time=28.9 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=55 time=11.9 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=55 time=46.1 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=55 time=20.6 ms
64 bytes from 14.215.177.39: icmp_seq=5 ttl=55 time=13.5 ms
64 bytes from 14.215.177.39: icmp_seq=6 ttl=55 time=13.2 ms
64 bytes from 14.215.177.39: icmp_seq=7 ttl=55 time=15.9 ms
64 bytes from 14.215.177.39: icmp_seq=8 ttl=55 time=18.4 ms
64 bytes from 14.215.177.39: icmp_seq=9 ttl=55 time=12.1 ms
^C
--- www.a.shifen.com ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8011ms
rtt min/avg/max/mdev = 11.975/20.108/46.115/10.518 ms
root@ALIENTEK:~#
```

图 3.22.1. 3 ping 百度测试上网功能

3.22.2 SoftAP（热点）模式

本次实验目的：测试 USB WIFI 开启热点，与手机连接

USER# source ./alientek_usb_wifi_setup.sh -m softap -d wlan0

```
root@ALIENTEK:~# source ./alientek_usb_wifi_setup.sh -m softap -d wlan0
您的WIFI配置信息是:
mode   : softap
device: wlan0
udhcpd (v1.24.1) started
Configuration file: //home/root/wifi.conf
drv->ifindex=6
l2_sock_recv==l2_sock_xmit=0x0x19[  972.313073] RTL871X: indicate disassoc
90640
+rtl871x_sta_deauth_ops, ff:ff:ff:ff:ff:ff is deauth, reason=2
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
Using interface wlan0 with hwaddr 00:13:ef:80:38:f7 and ssid 'alientek_softap'
rtl871x_set_wps_assoc_resp_ie
rtl871x_set_wps_beacon_ie
rtl871x_set_wps_probe_resp_ie
rtl871x_set_key_ops
rtl871x_set_beacon_ops
[  972.597585] RTL871X: set group key camid:1, addr:00:00:00:00:00:00, kid:1, type:TKIP
[  972.625815] RTL871X: assoc success
rtl871x_set_hidden_ssid ignore_broadcast_ssid:0, alientek_softap,15
rtl871x_set_acl
WIFI设置softap模式完成!
root@ALIENTEK:~#
```

图 3.22.2. 1 把 USB WIFI 设置成热点模式

打开手机设置，可以看到 USB WIFI 发出的热点，名称为“alientek_softap”，密码默认为12345678。输入密码后点击连接，即可连接到无线热点。



图 3.22.2. 2 手机上扫描 USB WIFI 热点名称



图 3.22.2. 3 输入密码并连接

若出现连接上了又断开的情况，用户请将连接 alientek_softap 热点信息设置为静态获取设置 IP 模式。如下图。注因手机厂商不一样，界面设置的方法可能不一样

IP 设置

静态

IP 地址	192.168.0.20
路由器	192.168.10.1
前缀长度	24
DNS 1	8.8.8.8
DNS 2	8.8.4.4

图 3.22.2. 4 设置静态 IP

手机连接 alientek_softap 热点信息时，串口终端打印的信息。

```
root@ALIENTEK:~# [  238.777730] RTL871X: send eapol packet
[  238.783852] RTL871X: set group key camid:1, addr:00:00:00:00:00:00, kid:1, type:TKIP
[  238.791749] RTL871X: recv eapol packet
[  238.796332] RTL871X: send eapol packet
[  238.809028] RTL871X: recv eapol packet
[  238.814452] RTL871X: set pairwise key camid:4, addr:60:ab:67:e5:93:e7, kid:0, type:AES
Sending OFFER of 192.168.0.20
Sending OFFER of 192.168.0.20
Sending OFFER of 192.168.0.20
Sending OFFER of 192.168.0.20
Sending ACK to 192.168.0.20
Sending ACK to 192.168.0.20
```

图 3.22.2. 5 连接过程驱动打印的信息

可以看到手机已经连接了热点，这样手机与开发板的 WIFI 构成了一个无线局域网。



图 3.22.2. 6 手机成功连接 USB WIFI 热点

如需要修改热点名称与热点密码，用户可以编辑脚本内容设置个人的热点信息。

```
#softap Mode（热点模式）
if [ "$mode" == "softap" ]; then
	processkill
	sleep 1
	sync
    echo -ne "interface=wlan0\nssid=alientek_softap\ndriver=rtl871xdrv\nchannel=6\nhw_mode=g\nignore_broadcast_ssid=0\n
auth_algs=1\nwpa=3\nwpa_passphrase=12345678\nwpa_key_mgmt=WPA-PSK\nwpa_pairwise=TKIP\nrsn_pairwise=CCMP" > $wifi_conf
	a=$(ifconfig |grep -E "br0" | grep -v grep | awk '{print $0}')
	if [ -n "$a" ];then
		brctl delif br0 $device
		brctl delif br0 $device
		ifconfig br0 down
	fi
	rm -rf /var/lib/misc/*
	touch /var/lib/misc/udhcpd.leases
	ifconfig $device up
	sleep 2
	ifconfig $device 192.168.1.38 netmask 255.255.255.0
	udhcpd -fS /etc/udhcpd.conf    &
	hostapd $wifi_conf -B
	export WIFI_MODE=softap
fi
```

图 3.22.2.7 查看脚本配置的热点名称及密码

3.22.3 Bridge（桥接）模式

本次实验目的：测试 USB WIFI 开启热点，并桥接到有线网络，让手机连接到热点并上网。ALPHA 开发板的网口请将网线插上其中一个或者两个，并且确保有线网络能上网。

用 ifconfig 查看有线网络 eth0 或者 eth1 能否获取 ip，并测试 eth0 是否能上网。下图是将网线插到开发板底板 ENET2 接口处，节点是 eth0，如果插到 ENET1 接口，节点是 eth1。MINI 底板只有一个网口，节点是 eth0。

```
root@ALIENTEK:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 8a:6c:b6:13:0b:c0
          inet addr:192.168.1.160  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::886c:b6ff:fe13:bc0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:972 errors:0 dropped:3 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69228 (67.6 KiB)  TX bytes:10875 (10.6 KiB)

eth1      Link encap:Ethernet  HWaddr 66:fe:1f:2b:d1:20
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1541 (1.5 KiB)  TX bytes:1541 (1.5 KiB)

root@ALIENTEK:~#
```

图 3.22.2.8 查看有线网络的节点

USER# source ./alientek_usb_wifi_setup.sh -m bridge -d wlan0 -e eth0

-e eth0 ：桥接的有线网络节点，根据实际情况桥接到对应的网络节点 eth0/eth1。

```
root@ALIENTEK:~# source ./alientek_usb_wifi_setup.sh -m bridge -d wlan0 -e eth0
您的WIFI配置信息是:
mode   : bridge
device: wlan0
ethernet  : eth0
[ 8506.034360] RTL871X: ERROR rtw_hal_macid_wakeup(wlan0): Invalid macid(32)
[ 8506.041598] RTL871X: ERROR rtw_hal_macid_wakeup(wlan0): Invalid macid(32)
[ 8506.058377] RTL871X: indicate disassoc
[ 8506.087596] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 8506.203046] fec 20b4000.ethernet eth0: Freescale FEC PHY driver [Generic PHY] (mii_bus:phy_addr=20b4000.ethernet:01, irq=-1)
[ 8508.303192] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 8509.203130] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[ 8509.211033] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[1]+  Killed                     udhcpd -fs /etc/udhcpd.conf
udhcpd (v1.24.1) started
udhcpd: is interface wlan0 up and configured?: Cannot assign requested address
[1]+  Done(1)                    udhcpd -fs /etc/udhcpd.conf
[ 8509.942031] device eth0 entered promiscuous mode
[ 8509.988546] device wlan0 entered promiscuous mode
[ 8510.035402] br0: port 1(eth0) entered forwarding state
[ 8510.040657] br0: port 1(eth0) entered forwarding state
Configuration file: //home/root/wifi.conf
drv->ifindex=6
+rtl871x_sta_deauth_ops, ff:ff:ff:ff:ff:ff is deauth, reason=2
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
Using interface wlan0 with hwaddr 00:13:ef:80:38:f7 and ssid 'alientek_bridge'
rtl871x_set_wps_assoc_resp_ie
rtl871x_set_wps_beacon_ie
rtl871x_set_wps_probe_resp_ie
rtl871x_set_key_ops
rtl871x_set_beacon_ops
[ 8510.354349] RTL871X: set group key camid:1, addr:00:00:00:00:00:00, kid:1, type:TKIP
[ 8510.376643] RTL871X: assoc success
[ 8510.381147] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
rtl871x_set_hidden_ssid ignore_broadcast_ssid:0, alientek_bridge,[ 8510.392309] br0: port 2(wlan0) entered forwarding state
15
[ 8510.400311] br0: port 2(wlan0) entered forwarding state
rtl871x_set_acl
WIFI设置bridge模式完成!
```

图 3.22.2.9 执行指令将无线网络 wlan0 桥接到 eth0

打开手机设置，可以看到 USB WIFI 发出的热点，名称为“alientek_bridge”，密码默认为 12345678。输入密码后点击连接，即可连接到无线热点。并测试能否上网，打开手机的浏览器测试即可。

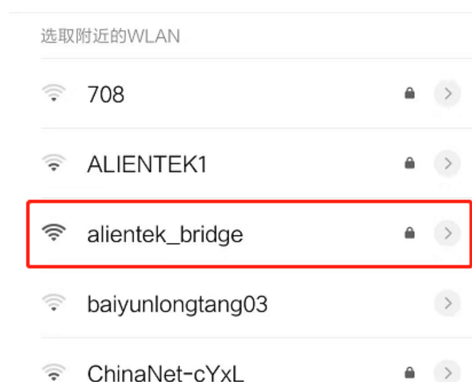


图 3.22.2. 10 查看 USB WIFI 的热点名称



图 3.22.2. 11 输入密码“12345678”后，连接成功示意图

连接时串口终端打印的信息：

```
[43163.774080] RTL871X: set pairwise key camid:4, addr:60:ab:67:e5:93:e7, kid:0, type:AES
[43296.717464] RTL871X: OnDeAuth(wlan0) reason=3, ta=60:ab:67:e5:93:e7
[43296.725943] RTL871X: clear key for addr:60:ab:67:e5:93:e7, camid:4
[43316.181079] RTL871X: send eapol packet
[43316.200608] RTL871X: recv eapol packet
[43316.205473] RTL871X: send eapol packet
[43316.217731] RTL871X: recv eapol packet
```

图 3.22.2. 12 连接成功打印驱动打印的信息

如需要修改热点名称与热点密码，用户可以编辑脚本内容设置个人的热点信息。

```
#bridge Mode(桥接模式)
if [ "$mode" == "bridge" ]; then
    ifconfig $device down
    ifconfig $device up
    ifconfig $ethernet down
    ifconfig $ethernet up
    sleep 2
    if [ "$WIFI_MODE" == "station" ]; then
        wpa_supplicant -B -D wext -i $device -c /etc/wpa_supplicant.conf
    fi
    processkill
    sleep 1
    sync
    echo -ne "interface=wlan0\nssid=alientek_bridge\ndriver=rtl871xdrv\nchannel=6\nhw_mode=g\nignore_broadcast_ssid=0\n
auth_algs=1\nwpa=3\nwpa_passphrase=12345678\nwpa_key_mgmt=WPA-PSK\nwpa_pairwise=TKIP\nrsn_pairwise=CCMP\nbridge=br0" > $wifi_conf
    rm -rf /var/lib/misc/*
    touch /var/lib/misc/udhcpd.leases
    udhcpd -fS /etc/udhcpd.conf &
    ifconfig $device 0.0.0.0
    brctl addbr br0
    ifconfig $ethernet 0.0.0.0
    brctl addif br0 $ethernet
    brctl addif br0 $device
    ifconfig br0 192.168.1.39 netmask 255.255.255.0
    hostapd $wifi_conf -B
    export WIFI_MODE=bridge
```

图 3.22.2. 13 脚本设置的热点名称与密码

3.23 ME3630-W 4G 模块测试

实验前准备 ME3630-W 4G 模块、天线和一张上网卡（联通 4G 卡或者移动 4G 卡）。

进行 4G 模块测试前，将移动或者联通 4G 卡插到底板的 SIM 卡槽里，再插上 ME3630-W 4G 模块，同时插上天线，天线接到模块的 MAIN 处。正确插入 4G 卡与天线后，开发板启动后底板上的 WWAN LED 会亮绿灯（WWAN LED 指示灯说明参考 3.18 小节），若此灯不亮，请检查 4G 卡是否插稳，ME3630-W 是否插稳，天线是否连接正确。ME3630-W 4G 模块安装如下图所示：



图 3.23. 1 ME3630 模块连接示意图

将 9、测试文件->shell->me3630-w 目录下的 6 个脚本文件拷贝到开发板文件系统的任意目录，本次拷贝到/home/root 目录下（开发板默认已经拷贝到时/home/root 目录下）。如下图

```
root@ALIENTEK-IMX6U:~# ls
disconnect  ecm-on  gosuncn_ecm_dialer  gosuncn_options  gosuncn_ppp_dialer  ppp-on
root@ALIENTEK-IMX6U:~#
```

图 3.23.2 拷贝 6 个脚本到文件系统目录下

这 6 个文件拷贝好以后要给予 disconnect、ppp-on、ecm-on 这 3 个 shell 脚本可执行权限。使用 chmod u+x +(脚本文件)赋予可执行权限。

```
root@ALIENTEK-IMX6U:~# chmod u+x ppp-on
root@ALIENTEK-IMX6U:~# chmod u+x ecm-on
root@ALIENTEK-IMX6U:~# chmod u+x disconnect
```

图 3.23.3 赋予脚本可执行权限

3.23.1 pppd 拨号上网

执行上一步拷贝的脚本开始 4G 网络连接

USER# ./ppp-on &

```
root@ALIENTEK-IMX6U:~# ./ppp-on &
```

图 3.23.1.1 执行脚本进行拨号上网

```
pppd options in effect:
debug           # (from gosuncn_options)
nodetach                # (from gosuncn_options)
persist         # (from gosuncn_options)
dump            # (from gosuncn_options)
noauth          # (from gosuncn_options)
user Anyname            # (from gosuncn_options)
password ??????         # (from gosuncn_options)
/dev/ttyUSB2            # (from gosuncn_options)
115200          # (from gosuncn_options)
lock            # (from gosuncn_options)
connect chat -v -f /home/root/gosuncn_ppp_dialer                # (from command line)
crtscts         # (from gosuncn_options)
modem           # (from gosuncn_options)
novj            # (from gosuncn_options)
ipcp-accept-local               # (from gosuncn_options)
ipcp-accept-remote              # (from gosuncn_options)
noipdefault             # (from gosuncn_options)
defaultroute            # (from gosuncn_options)
usepeerdns              # (from gosuncn_options)
noccp           # (from gosuncn_options)
nobsdcomp               # (from gosuncn_options)
ATE
OK
ATH
OK
ATP
OK
AT+CGDCONT=1,"IP","3GNET"
OK
ATD*99#
CONNECT
Script chat -v -f /home/root/gosuncn_ppp_dialer finished (pid 737), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x621440f2> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x9d9955dc> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x9d9955dc> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x621440f2> <pcomp> <accomp>]
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <addr 10.170.104.128>]
sent [IPCP ConfAck id=0x1 <addr 10.170.104.128>]
rcvd [IPV6CP ConfReq id=0x1 <addr fe80::d1a2:2011:2faa:f6ed>]
Unsupported protocol 'IPv6 Control Protocol' (0x8057) received
sent [LCP ProtRej id=0x2 80 57 01 01 00 0e 01 0a d1 a2 20 11 2f aa f6 ed]
rcvd [IPCP ConfNak id=0x1 <addr 10.170.104.127> <ms-dns1 221.179.38.7> <ms-dns2 221.179.38.7>]
sent [IPCP ConfReq id=0x2 <addr 10.170.104.127> <ms-dns1 221.179.38.7> <ms-dns2 221.179.38.7>]
rcvd [IPCP ConfAck id=0x2 <addr 10.170.104.127> <ms-dns1 221.179.38.7> <ms-dns2 221.179.38.7>]
Failed to create /etc/ppp/resolv.conf: No such file or directory
not replacing existing default route via 192.168.1.1
local  IP address 10.170.104.127
remote IP address 10.170.104.128
primary   DNS address 221.179.38.7
secondary DNS address 221.179.38.7

root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 2 拨号连接成功打印的信息

使用 ifconfig 指令查看获取的 ip 地址：

USER# ifconfig

```
root@ALIENTEK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 66:4c:fe:80:44:f8
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 3a:b9:a7:66:98:4a
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:66 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4556 (4.4 KiB)  TX bytes:4556 (4.4 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.197.179.1  P-t-P:10.197.179.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:68 (68.0 B)  TX bytes:54 (54.0 B)

usb0      Link encap:Ethernet  HWaddr 26:2b:a2:18:ba:11
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 3 查看获取的 ip 地址

通过 ping www.baidu.com 来测试是否能上网，为了测试准确，请把其他的网络接口拔出或者使用 ifconfig 指令将其他网卡关掉。

```
root@ALIENTEK-IMX6U:~# ifconfig eth0 down
root@ALIENTEK-IMX6U:~# ifconfig eth1 down
root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 4 关闭有线网卡

关闭上面的网口后执行 ping www.baidu.com 出现识别不了百度主机的情况，请使用 vi 指令编辑 /etc/resolv.conf 这个文件，将 nameserver 127.0.0.1 改为 8.8.8.8 或者 114.114.114.114（DNS 域名服务器地址，由这个地址去解释百度地址）。

```
root@ALIENTEK-IMX6U:~# ping www.baidu.com
ping: unknown host www.baidu.com
root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 5 修改 DNS 服务器地址

USER# vi /etc/resolv.conf

修改如下图：

```
# Generated by Connection Manager
nameserver 8.8.8.8
nameserver ::1
```

图 3.23.1. 6 修改 DNS 示意图

再执行 ping www.bai.com 指令，看到有数据回复，如下图，表明能正常上网

USER# ping www.baidu.com -I ppp0　　// “-I”参数是指定网卡名

```
root@ALIENTEK-IMX6U:~# vi /etc/resolv.conf
root@ALIENTEK-IMX6U:~# ping www.baidu.com -I ppp0
PING www.wshifen.com (103.235.46.39) from 10.197.179.1 ppp0: 56(84) bytes of data.
64 bytes from 103.235.46.39: icmp_seq=1 ttl=46 time=50.0 ms
64 bytes from 103.235.46.39: icmp_seq=2 ttl=47 time=40.2 ms
64 bytes from 103.235.46.39: icmp_seq=3 ttl=47 time=39.7 ms
64 bytes from 103.235.46.39: icmp_seq=4 ttl=46 time=36.4 ms
^C64 bytes from 103.235.46.39: icmp_seq=5 ttl=45 time=40.2 ms

--- www.wshifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 36.438/41.347/50.076/4.592 ms
root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 7 ping 百度上网测试

断开连接请执行 disconnect 脚本。

USER# ./disconnect

```
root@ALIENTEK-IMX6U:~# ./disconnect
Terminating on signal 15
Connect time 0.2 minutes.
Sent 0 bytes, received 0 bytes.
sent [LCP TermReq id=0x3 "User request"]
root@ALIENTEK-IMX6U:~# rcvd [LCP TermAck id=0x3]
Connection terminated.

[1]+  Done(5)                 ./ppp-on
root@ALIENTEK-IMX6U:~#
```

图 3.23.1. 8 断开连接测试

3.23.2 通过 ECM 上网

要配置 ECM 模式上网，请先执行 disconnect 脚本断开 pppd 拨号上网，再执行下面的指令配置成 ECM 模式链接网络。

USER# ./ecm-on &

```
pppd options in effect:
debug           # (from gosuncn_options)
nodetach                # (from gosuncn_options)
persist         # (from gosuncn_options)
dump            # (from gosuncn_options)
noauth          # (from gosuncn_options)
user Anyname            # (from gosuncn_options)
password ??????         # (from gosuncn_options)
/dev/ttyUSB2            # (from gosuncn_options)
115200          # (from gosuncn_options)
lock            # (from gosuncn_options)
connect chat -v -f /home/root/gosuncn_ecm_dialer                # (from command line)
crtscts         # (from gosuncn_options)
modem           # (from gosuncn_options)
novj            # (from gosuncn_options)
ipcp-accept-local               # (from gosuncn_options)
ipcp-accept-remote              # (from gosuncn_options)
noipdefault             # (from gosuncn_options)
defaultroute            # (from gosuncn_options)
usepeerdns              # (from gosuncn_options)
noccp           # (from gosuncn_options)
nobsdcomp               # (from gosuncn_options)
ATE
OK
ATH
OK
ATP
OK
AT+ZSWITCH=L
OK
AT+CGDCONT=1,"IP","CMNET"
OK
AT+ZECMCALL=1[  103.131727] IPv6: ADDRCONF(NETDEV_CHANGE): usb0: link becomes ready
[  103.138113] cdc_ether 2-1.2:1.3 usb0: kevent 12 may have been dropped

+ZECMCALL: CONNECT
OK
ATD*99#
CONNECT
Script chat -v -f /home/root/gosuncn_ecm_dialer finished (pid 757), status = 0x0
Serial connection established.
using channel 2
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
Modem hangup
Connection terminated.

root@ALIENTEK-IMX6U:~#
```

图 3.23.2. 1 配置 ECM 模式上网

使用 ifconfig 指令查看获取的 ip 地址。

USER# ifconfig

```
root@ALIENTEK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 36:28:6f:07:7b:a9
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 4e:9a:57:d4:ea:7f
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:66 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4556 (4.4 KiB)  TX bytes:4556 (4.4 KiB)

usb0      Link encap:Ethernet  HWaddr 36:da:2d:4f:4b:e6
          inet addr:10.4.154.13  Bcast:10.4.154.15  Mask:255.255.255.252
          inet6 addr: fe80::34da:2dff:fe4f:4be6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:59 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3226 (3.1 KiB)  TX bytes:10221 (9.9 KiB)

root@ALIENTEK-IMX6U:~#
```

图 3.23.2. 2 查看获取的 ip 地址

为了准确测试，测试前请把其他网口网线拔出或者关掉。使用 ifconfig 指令关掉其他网卡。

```
root@ALIENTEK-IMX6U:~# ifconfig eth0 down
root@ALIENTEK-IMX6U:~# ifconfig eth1 down
root@ALIENTEK-IMX6U:~#
```

图 3.23.2. 3 关闭有线网络网卡

执行 ping www.bai.com 指令，看到有数据回复，如下图，表明能正常上网

USER# ping www.baidu.com -I usb0　　// “-I”参数是指定网卡名

```
root@ALIENTEK-IMX6U:~# ping www.baidu.com -I usb0
PING www.a.shifen.com (183.232.231.172) from 10.4.154.13 usb0: 56(84) bytes of data.
64 bytes from 183.232.231.172: icmp_seq=1 ttl=55 time=31.0 ms
64 bytes from 183.232.231.172: icmp_seq=2 ttl=55 time=28.2 ms
64 bytes from 183.232.231.172: icmp_seq=3 ttl=55 time=22.4 ms
64 bytes from 183.232.231.172: icmp_seq=4 ttl=55 time=32.5 ms
64 bytes from 183.232.231.172: icmp_seq=5 ttl=55 time=32.2 ms
64 bytes from 183.232.231.172: icmp_seq=6 ttl=55 time=26.9 ms
^C
--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 22.438/28.897/32.513/3.532 ms
root@ALIENTEK-IMX6U:~#
```

图 3.23.2. 4 ping 百度测试上网

3.24 SDIO WIFI 测试

实验前准备正点原子 SDIO WIFI 模块（RTL8189）。

由于底板 SDIO WIFI 接口与 SD 卡槽接口共用了大部分管脚，所以在使用 SDIO WIFI 时不要使用 SD 卡。也就是说，我们测试 SDIO WIFI 时不能从 SD 卡启动系统。我们应该从 eMMC 或者 Nand Flash 启动系统。

本次实验我们从 eMMC 启动系统，启动系统前，请将 SDIO WIFI 模块插进 SDIO 接口处，如下图所示：

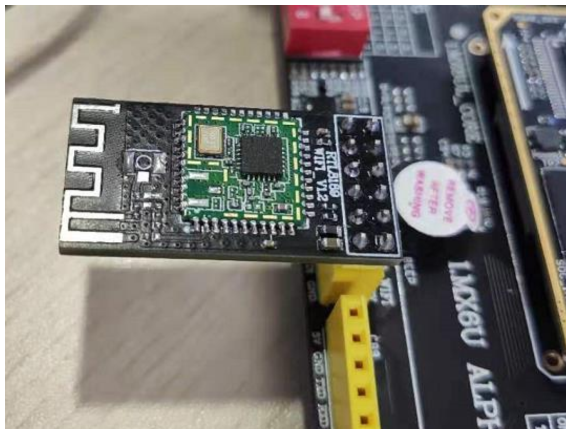


图 3.24. 1 SDIO WIFI 接法示意图

将 9、测试文件->driver->rtl8189 下的 8189fs.ko 驱动模块文件拷贝到开发板文件系统的任意目录，本次拷贝到/home/root 目录下（开发板默认已经拷贝到时/home/root 目录下）。如下图

```
root@ALIENTEK-IMX6U:~# ls
8189fs.ko
root@ALIENTEK-IMX6U:~#
```

图 3.24. 2 拷贝驱动文件到文件系统目录下

使用 insmod 指令安装此模块

USER# insmod 8189fs.ko

```
root@ALIENTEK-IMX6U:~# insmod 8189fs.ko
[  159.621243] RTL871X: module init start
[  159.627806] RTL871X: rtl8189fs v4.3.24.8_22657.20170607
[  159.694310] RTL871X: HW EFUSE
[  159.697333] RTL871X: hal_com_config_channel_plan chplan:0x20
[  159.923602] RTL871X: rtw_regsty_chk_target_tx_power_valid return _FALSE for band:0, path:0, rs:0, t:-1
[  159.943293] RTL871X: rtw_ndev_init(wlan0) if1 mac_addr=30:4a:26:b1:45:62
[  159.964050] RTL871X: module init ret=0
root@ALIENTEK-IMX6U:~#
```

图 3.24. 3 安装驱动文件

使用 rfkill list 指令查看射频是否开启了，blocked 是关闭的意思，可以看到 Soft blocked 的状态是被关闭了。

USER# rfkill list

```
root@ALIENTEK-IMX6U:~# rfkill list
0: phy0: wlan
        Soft blocked: yes
        Hard blocked: no
root@ALIENTEK-IMX6U:~#
```

图 3.24. 4 查看射频是否开启

所以使用 rfkill unblock all 指令来解锁设备（注：如果上面没有被关闭，则不用解锁），如下图红色框位置，有两个 no，则表示已经解锁。

USER# rfkill unblock all

```
root@ALIENTEK-IMX6U:~# rfkill list
0: phy0: wlan
        Soft blocked: yes
        Hard blocked: no
root@ALIENTEK-IMX6U:~# rfkill unblock all
root@ALIENTEK-IMX6U:~# [   45.760547] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready

root@ALIENTEK-IMX6U:~# rfkill list
0: phy0: wlan
        Soft blocked: no
        Hard blocked: no
root@ALIENTEK-IMX6U:~#
```

图 3.24. 5 解锁射频

扫描附近无线网络信息并打印，如下图扫描到本公司的无线网络名称“ALIENTEK”，还能看到加密类型等。（备注：还可以用其他指令来扫描无线网络信息，如 iwlist wlan0 scan）

USER# wpa_cli -i wlan0 scan_result

```
root@ALIENTEK-IMX6U:~# wpa_cli -i wlan0 scan_result
bssid / frequency / signal level / flags / ssid
48:ee:0c:44:c7:ca       2412    100     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Stone123456
88:f8:72:8d:f3:18       2437    100     [WPA2-PSK-CCMP][WPS][ESS]       ZZK
00:6b:8e:fb:af:50       2462    100     [WPA2-PSK-CCMP][ESS]    PDCN
88:f8:72:8d:f3:19       2437    100     [WPA2-PSK-CCMP][ESS]
e4:0e:ee:f2:11:19       2462    86      [WPA2-PSK-CCMP][ESS]
bc:5f:f6:a5:f5:9f       2472    90      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      FAE
e4:0e:ee:f2:11:14       2462    84      [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][WPS][ESS]       ALIENTEK
e4:0e:ee:f2:11:15       2462    96      [WPA2-PSK-CCMP][ESS]
0c:4b:54:1f:04:1a       2462    56      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      708
root@ALIENTEK-IMX6U:~#
```

图 3.24. 6 扫描附近无线网络信息

在当前目录下编辑一个 wpa_supplicant.conf 配置文件（这里说明一下：在系统/etc/下也有一个 wpa_supplicant.conf，系统在安装此模块后会自动使用/etc/wpa_supplicant.conf 的无线网络账号与密码来配置网络，我们需要自己使用指令来配置使用无线网络，否则内核会打印一些报错）。

将下面的内容在当前/home/root/目录下编辑成一个 wpa_supplicant.conf 配置文件。

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="ALIENTEK"
        psk="15902020353"
}
```

解释：

- ssid 为无线网络名称
- psk 为无线网络密码

编辑完成后在当前目录有 wpa_supplicant.conf 这个配置文件，用来配置无线网络上网。

```
root@ALIENTEK-IMX6U:~# ls
8189fs.ko  wpa_supplicant.conf
root@ALIENTEK-IMX6U:~#
```

图 3.24. 7 编辑的 wpa_supplicant.conf 配置文件

在安装 wifi 模块后，系统会启动 wpa_supplicant 相关进程来自动连接网络，我们要先关闭这个进程，才能重新配置。

USER# killall wpa_supplicant

```
root@ALIENTEK-IMX6U:~# killall wpa_supplicant
root@ALIENTEK-IMX6U:~# [  115.533634] RTL871X: set bssid:00:00:00:00:00:00
```

图 3.24.8 杀死自动开启的 wpa_supplicant 相关进程

使用下面的指令来连接配置的无线网络，当看到“wlan0: CTRL-EVENT-CONNECTED - Connection to ec:88:8f:77:d6:da completed [id=0 id_str=]”这句话表明连接成功。如果连接不成功会提示“wlan0: WPA: 4-Way Handshake failed - pre-shared key may be incorrect”这样的话，请用户检查无线网络信息是否有误。

USER# wpa_supplicant -Dnl80211 -c wpa_supplicant.conf -i wlan0 &

```
root@ALIENTEK-IMX6U:~# wpa_supplicant -Dnl80211 -c wpa_supplicant.conf -i wlan0 &
[1] 708
root@ALIENTEK-IMX6U:~# Successfully initialized wpa_supplicant
[  149.839653] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
wlan0: Trying to associate with e[  151.520159] RTL871X: rtw_set_802_11_connect(wlan0)  fw_state=0x00000008
4:0e:ee:f2:11:14 (SSID='ALIENTEK' freq=2462 MHz)
[  151.603211] RTL871X: start auth
[  151.609259] RTL871X: auth success, start assoc
[  151.622371] RTL871X: assoc success
[  151.626079] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[  151.632799] RTL871X: recv eapol packet
wlan0: Associated with e4:0e:ee:f[  151.649013] RTL871X: send eapol packet
2:11:14
[  151.683465] RTL871X: recv eapol packet
[  151.690908] RTL871X: send eapol packet
[  151.696298] RTL871X: set pairwise key camid:4, addr:e4:0e:ee:f2:11:14, kid:0, type:AES
wlan0: WPA: Key negotiation completed with e4:0e:ee:f2:11:14 [PTK[  151.709553] RTL871X: set group key camid:5, addr:e4:0e:ee:f2:11:14, kid:1, type:TKIP
=CCMP GTK=TKIP]
wlan0: CTRL-EVENT-CONNECTED - Connection to e4:0e:ee:f2:11:14 completed [id=0 id_str=]

root@ALIENTEK-IMX6U:~#
```

图 3.24.9 连接无线网络热点

连接成功后，为 wlan0 获取 ip

USER# udhcpc -i wlan0

```
root@ALIENTEK-IMX6U:~# udhcpc -i wlan0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 192.168.2.84...
Lease of 192.168.2.84 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.2.1
root@ALIENTEK-IMX6U:~#
```

图 3.24.10 为 wlan0 获取 ip

为了准确测试，测试前请把其他网口网线拔出或者关掉。使用 ifconfig 指令关掉其他网卡。

```
root@ALIENTEK-IMX6U:~# ifconfig eth0 down
root@ALIENTEK-IMX6U:~# ifconfig eth1 down
root@ALIENTEK-IMX6U:~#
```

图 3.24.11 关闭有线网络的网卡

执行下面的指令测试上网，参数“-I”指定网卡。这里要指定 wlan0 网卡。可按“Ctrl + c”终止 ping 指令。出现如下结果，说明 ping 百度成功。

```
root@ALIENTEK-IMX6U:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.2.84 wlan0: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=55 time=11.2 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=55 time=12.5 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=55 time=12.7 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=55 time=8.64 ms
64 bytes from 14.215.177.39: icmp_seq=5 ttl=55 time=20.6 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 8.640/13.158/20.607/4.002 ms
root@ALIENTEK-IMX6U:~#
```

图 3.24.12 ping 百度测试联网功能

第四章 ALIENTEK I.MX6U 交叉编译

开发环境的搭建是开发人员中很重要的一步，我们将极大简化您的开发环境。为此我们向您提供了交叉编译工具链。这个编译工具链是 yocto DISTRO_VERSION 4.1.15.2.1.0 版本编译出来的 sdk 工具包，已经经过裁剪和一定程度的优化与测试，目的是为了在使用过程中减少出错。编译工具链已经封装成一个脚本文件 fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh，可直接执行该脚本进行安装，安装之后使能环境变量就可以用来编译内核、U-boot、Qt 工程，还可以用来单独交叉编译 c 文件生成可以执行的二进制文件。

4.1 版本说明

主机版本：Ubuntu14.04 以上版本
交叉工具链版本为：arm-poky-linux-gnueabi-gcc (GCC) 5.3.0
U-boot 版本：uboot-imx-2016.03-2.1.0
内核版本：linux-imx-4.1.15-2.1.0
Qt 版本：5.6.2

Qt 版本特别说明：我们文件系统里使用的 Qt 版本为 Qt5.6.2。其中我们向文件系统添加了 QWebKit 和使 Qt 交叉编译工具也支持 QWebKit 编译。官方版本 Qt5.6 以后去除了 QWebKit，取而代之的是 QWebEngine（本文件系统不支持），而 QWebEngine 在 windows 下编程接口暂时不丰富，Qt5.6 的 MinGW 版本的 Qt 不支持 QWebEngine，仅仅支持 MSVC 版本。如果想使用 QWebKit 在 Windows 下编程或者在 Linux 下编程的小伙伴请使用我们 Qt 教程（敬请期待！）里面的 Qt5.5.1 版本。Qt5.5.1 下的工程可以直接使用我们的提供的这个工具链编译就可以编译 QWebKit 相关工程，然后拷贝可执行文件到我们开发板上运行。

4.2 搭建交叉编译环境

把->开发板光盘->5、开发工具->1、交叉编译器->fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh（yocto 编译出来的 sdk 工具包）拷贝到 Ubuntu 虚拟机(可以使用 Winscp、FileZilla 或建立 Samba 搭建文件共享的方式拷贝到 Ubuntu 虚拟机，本次认为您已经拷贝了该脚本到 Ubuntu 虚拟机)。

如下图本文已经把交叉编译工具拷贝到了 Ubuntu 虚拟机。

```
alientek@ubuntu:~$ ls
fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh
alientek@ubuntu:~$
```

图 4.2. 1 拷贝 sdk 工具包到 ubuntu 系统里

执行下面的指令修改脚本的权限，并且可以看到当执行完成下指令后，脚本在终端所显示的颜色也发生了相应的变化。

USER# chmod u+x fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh

```
alientek@ubuntu:~$ chmod u+x fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh
alientek@ubuntu:~$ ls
fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh  http  Pictures  QtCourse  QW
alientek@ubuntu:~$
```

图 4.2. 2 赋予脚本可执行权限

直接执行脚本安装交叉编译工具，连续敲下两次回车键确认，再输入用户密码即可。本次安装的目录为脚本所指定的默认安装的目录，后面的内核编译环境的交叉编译都是按这个安装目录去操作，所以建议您也是默认安装到/opt/fsl-imx-x11/4.1.15-2.1.0 这个默认目录。

```
alientek@ubuntu:~$ ./fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh
Freescale i.MX Release Distro SDK installer version 4.1.15-2.1.0
================================================================
Enter target directory for SDK (default: /opt/fsl-imx-x11/4.1.15-2.1.0):
You are about to install the SDK to "/opt/fsl-imx-x11/4.1.15-2.1.0". Proceed[Y/n]?
[sudo] password for alientek:
Extracting SDK...............................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
 $ . /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~$
```

图 4.2.3 安装 sdk 工具到默认的目录

使用方法也十分简单，根据上面打印出来的提示，直接使能环境变量就可以了。但是在不同终端或者切换用户时需要重新使能环境变量方可使用。

USER# source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

```
alientek@ubuntu:~$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~$
```

图 4.2.4 使能环境变量

使能环境变量后可以使用 env 指令查看生效的环境变量，下图为部分截图，可以看出使能了这个环境变量后 gcc 已经配置好编译时所用的参数，如硬浮点参数-mfpu=neon -mfloat-abi=hard。使用硬浮点交叉编译，可以使用 CPU 自带 FPU。

USER# env

```
OE_QMAKE_LINK=arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
GDM_LANG=zh_CN
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
QMAKESPEC=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++
QT_CONF_PATH=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qt.conf
LC_MEASUREMENT=en_US.UTF-8
OECORE_NATIVE_SYSROOT=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux
COMPIZ_CONFIG_PROFILE=ubuntu
IM_CONFIG_PHASE=1
CONFIGURE_FLAGS=--target=arm-poky-linux-gnueabi --host=arm-poky-linux-gnueabi --build=x86_64-linux --with-libtool-sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
PAPERSIZE=letter
GDMSESSION=ubuntu
OE_QMAKE_QT_CONFIG=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/qconfig.pri
OE_QMAKE_CC=arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
KCFLAGS=--sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
CXX=arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
OBJCOPY=arm-poky-linux-gnueabi-objcopy
SESSIONTYPE=gnome-session
SHLVL=1
HOME=/home/alientek
XDG_SEAT=seat0
LANGUAGE=zh_CN:en_US:en
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
UPSTART_INSTANCE=
UPSTART_EVENTS=started starting
LOGNAME=alientek
OE_QMAKE_MOC=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/moc
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-jSJFaWS191
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/
QT4_IM_MODULE=fcitx
OE_QMAKE_QDBUSCPP2XML=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qdbuscpp2xml
OECORE_ACLOCAL_OPTS=-I /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/share/aclocal
PKG_CONFIG_PATH=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/pkgconfig
LESSOPEN=| /usr/bin/lesspipe %s
ARCH=arm
RANLIB=arm-poky-linux-gnueabi-ranlib
OE_QMAKE_CFLAGS=
INSTANCE=Unity
TEXTDOMAIN=im-config
UPSTART_JOB=unity-settings-daemon
CROSS_COMPILE=arm-poky-linux-gnueabi-
CC=arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
XDG_RUNTIME_DIR=/run/user/1000
```

图 4.2.5 查看使能后的环境变量

使用 arm-poky-linux-gnueabi-gcc -v 指令可以查看 gcc 版本，表明环境变量已经生效。

USER# arm-poky-linux-gnueabi-gcc --version

```
alientek@ubuntu:~$ arm-poky-linux-gnueabi-gcc --version
arm-poky-linux-gnueabi-gcc (GCC) 5.3.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

alientek@ubuntu:~$
```

图 4.2.6 查看 gcc 版本信息

经过实验 ubuntu14.04 内核编译时会出现/bin/sh: 1: lzop: not found 提示错误。所以我们只需要安装缺少的东西即可。

首先我们先更新软件列表

USER# sudo apt-get update

USER# sudo apt-get install lzop

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ sudo apt-get install lzop
[sudo] password for alientek:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
  lzop
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 435 个软件包未被升级。
需要下载 43.4 kB 的软件包。
解压缩后会消耗掉 118 kB 的额外空间。
获取：1 http://mirrors.aliyun.com/ubuntu/ trusty/universe lzop amd64 1.03-3 [43.4 kB]
下载 43.4 kB，耗时 0秒 (227 kB/s)
正在选中未选择的软件包 lzop。
(正在读取数据库 ... 系统当前共安装有 180064 个文件和目录。)
正准备解包 .../archives/lzop_1.03-3_amd64.deb  ...
正在解包 lzop (1.03-3) ...
正在处理用于 man-db (2.6.7.1-1ubuntu1) 的触发器 ...
正在设置 lzop (1.03-3) ...
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.2.7 安装相应的软件包

要显示 menucofig 菜单时需要安装下面的库，执行下面的指令

USER# sudo apt-get install libncurses*

按如上步骤搭建完成环境变量就可以使用这个交叉编译工具来编译内核、U-boot 和 Qt 工程了。

4.3 编译内核

将 1、例程源码->3、正点原子修改后的 Uboot 和 Linux->linux-imx-4.1.15-2.1.0-gxxxxxx.tar-vx.x.bz2 内核源码解压到 Ubuntu 拷贝到虚拟机 Ubuntu 目录下，使用 tar 指令解压，格式是 tar jvxf + 内核源码压缩包。

解压后，进入内核源码目录如下：

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ ls
arch   COPYING  crypto         drivers   fs       init  Kbuild   kernel  MAINTAINERS  mm   README           samples  security  tools  virt
block  CREDITS  Documentation  firmware  include  ipc   Kconfig  lib     Makefile     net  REPORTING-BUGS   scripts  sound           usr
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.3 1 ls 指令查看内核源码目录下的文件

编辑编译脚本，使用 vi/vim 指令编辑一个名称为 build.sh 的脚本，拷贝以下内容到该脚本。按 i 进入插入模式，右键进行粘贴内容，粘贴成功后按 ESC 键退出编辑模式，然后输入:wq 保存并退出。

USER# vi build.sh

#!/bin/bash

#清理内核
make distclean
#内核配置，参数-j 16 的意思为同时让 make 最多允许 16 条编译指令执行
make imx_v7_defconfig -j 16
#编译内核
make zImage -j 16
#编译 emmc 设备树，含正点原子 RGB 屏全部分辨率
make imx6ull-14x14-emmc-10.1-1280x800-c.dtb
make imx6ull-14x14-emmc-7-1024x600-c.dtb
make imx6ull-14x14-emmc-7-800x480-c.dtb
make imx6ull-14x14-emmc-4.3-800x480-c.dtb
make imx6ull-14x14-emmc-4.3-480x272-c.dtb
#编译 nandflash 设备树，含正点原子 RGB 屏全部分辨率
make imx6ull-14x14-nand-10.1-1280x800-c.dtb
make imx6ull-14x14-nand-7-1024x600-c.dtb
make imx6ull-14x14-nand-7-800x480-c.dtb
make imx6ull-14x14-nand-4.3-800x480-c.dtb
make imx6ull-14x14-nand-4.3-480x272-c.dtb

```
alientek@ubuntu: ~/IMX6/linux-imx-4.1.15-2.1.0
#!/bin/bash
make distclean
make imx_v7_defconfig -j 16
make zImage -j 16

make imx6ull-14x14-emmc-10.1-1280x800-c.dtb
make imx6ull-14x14-emmc-7-1024x600-c.dtb
make imx6ull-14x14-emmc-7-800x480-c.dtb
make imx6ull-14x14-emmc-4.3-800x480-c.dtb
make imx6ull-14x14-emmc-4.3-480x272-c.dtb

make imx6ull-14x14-nand-10.1-1280x800-c.dtb
make imx6ull-14x14-nand-7-1024x600-c.dtb
make imx6ull-14x14-nand-7-800x480-c.dtb
make imx6ull-14x14-nand-4.3-800x480-c.dtb
make imx6ull-14x14-nand-4.3-480x272-c.dtb
~
~
~
~
~
~
~
:wq
```

图 4.3 2 编辑编译脚本

用 ls 命令可以看到我们的 build.sh 脚本。

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ ls
arch      CREDITS        firmware  ipc      lib          net             scripts   usr
block     crypto         fs        Kbuild   MAINTAINERS  README          security  virt
build.sh  Documentation  include   Kconfig  Makefile     REPORTING-BUGS  sound
COPYING   drivers        init      kernel   mm           samples         tools
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.3 3 用 ls 查看编译脚本

使用 chmod 指令修改 build.sh 的权限，修改完成权限后，可以看到 build.sh 的终端显示颜色也随之发生变化。

USER# chmod u+x build.sh

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ chmod u+x build.sh
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ ls
arch      CREDITS        firmware  ipc      lib          net            scripts   usr
block     crypto         fs        Kbuild   MAINTAINERS  README         security  virt
build.sh  Documentation  include   Kconfig  Makefile     REPORTING-BUGS sound
COPYING   drivers        init      kernel   mm           samples        tools
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.3 4 赋予编译脚本可执行权限

使用 source 指令使能当前终端交叉编译环境变量（切换终端和用户需要重新执行下面这条指令）。

USER# source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.3 5 使能环境变量

直接在当前路径下执行该脚本进行编译内核与多个设备树。

USER# ./build.sh

下面为编译内核的部分截图

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ source build.sh
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
scripts/kconfig/conf  --silentoldconfig Kconfig
```

图 4.3 6 编译内核过程的部分截图

编译成功如下图所示，可以看到 zImage 与各个设备树已经编译好了。同时也看到它们所在的路径，例如内核 zImage 就在 arch/arm/boot/目录下。

```
  Kernel: arch/arm/boot/zImage is ready
  DTC     arch/arm/boot/dts/imx6ull-14x14-emmc-10.1-1280x800-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-emmc-7-1024x600-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-emmc-7-800x480-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-emmc-4.3-800x480-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-emmc-4.3-480x272-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-nand-10.1-1280x800-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-nand-7-1024x600-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-nand-7-800x480-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-nand-4.3-800x480-c.dtb
  DTC     arch/arm/boot/dts/imx6ull-14x14-nand-4.3-480x272-c.dtb
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.3 7 编译后的设备树与内核路径

更新内核与设备树的方法比较简单直接把内核设备树复制到 SD 卡系统卡的第一个分区

“boot”下替换即可，如果是 emmc 设备启动，复制到/dev/mmcblk1p1 目录下替换相应的文件。如果是 nandflash 设备启动需要重新烧写 nandflash 的内核与设备树的分区。

4.4 编译内核模块

在这里说明下，重新编译内核模块不是必须的，如果您有改动过内核模块选项，则需要重新编译内核。

前提条件：先按照上面步骤编译出内核才可以编译内核模块，确保当前交叉编译工具链已经在当前终端环境生效。如果没生效需要执行 source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi 指令。

执行下面指令编译出内核模块

USER# make modules -j 16

编译完成如下图

```
  LD [M]  drivers/usb/gadget/legacy/gadgetfs.ko
  LD [M]  drivers/usb/gadget/libcomposite.ko
  LD [M]  drivers/usb/serial/ftdi_sio.ko
  LD [M]  drivers/usb/serial/option.ko
  LD [M]  drivers/usb/serial/usb_wwan.ko
  LD [M]  drivers/usb/serial/usbserial.ko
  LD [M]  drivers/video/fbdev/mxc/mxc_dcic.ko
  LD [M]  fs/binfmt_misc.ko
  LD [M]  fs/configfs/configfs.ko
  LD [M]  fs/fat/msdos.ko
  LD [M]  fs/isofs/isofs.ko
  LD [M]  fs/nls/nls_iso8859-15.ko
  LD [M]  fs/udf/udf.ko
  LD [M]  lib/crc-ccitt.ko
  LD [M]  lib/crc-itu-t.ko
  LD [M]  lib/libcrc32c.ko
  LD [M]  lib/crc7.ko
  LD [M]  sound/core/snd-hwdep.ko
  LD [M]  sound/core/snd-rawmidi.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/snd-usb-audio.ko
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.4.1 编译内核模块

SD 卡更新内核模块的方法：

将 SD 卡系统卡用读卡器插入到 Ubuntu 虚拟机，等待 Ubuntu 读出 SD 卡系统卡读出系统分区并挂载分区如下图。两个分区分别为‘rootfs’分区与‘boot’分区。我们内核模块在 lib/modules 目录下。

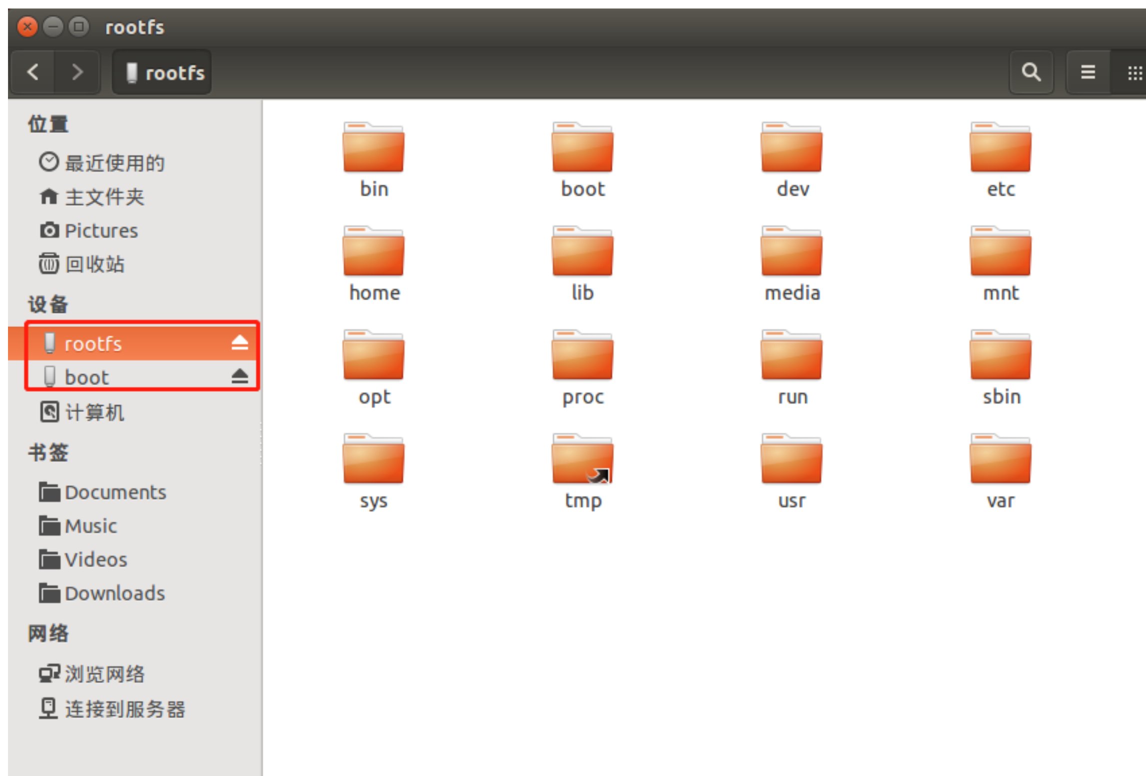


图 4.4. 2 ubuntu 识别 SD 卡系统卡的分区

使用 ls 命令查看 rootfs 目录挂载的位置，一般地会挂载在/media/ + ‘您的用户名’下。

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ ls /media/alientek/rootfs/
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  run  sbin  sys  tmp  usr  var
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$
```

图 4.4. 3 查看 SD 卡系统卡的挂载目录

若要安装模块到挂载的目录下，需要更多的权限，我们需要切换到 root 用户下（root 用户目录是最高权限的用户，代表着“大佬”的身份），如果您的开发环境还没有添加 root 用户，可以按如下的方法进行创建并切换到 root 用户。

```
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ sudo passwd root
输入新的 UNIX 密码：
重新输入新的 UNIX 密码：
passwd：已成功更新密码
alientek@ubuntu:~/IMX6/linux-imx-4.1.15-2.1.0$ su root
密码：
root@ubuntu:/home/alientek/IMX6/linux-imx-4.1.15-2.1.0#
```

图 4.4. 4 添加 root 用户及设置密码

比如我已经确认挂载的位置就是/media/alientek/rootfs，确认后执行下面指令安装模块到 rootfs 目录下。

按照上面切换到 root 用户，在上面已经说过了，当切换终端或者用户都需要重新使能环境变量。

执行 source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi 指令重新使能环境变量，再执行下面的指令安装内核模块，默认会安装在指定目录下的 lib/modules 目录下。

USER# make modules_install INSTALL_MOD_PATH=/media/alientek/rootfs

安装成功如下图，这个就是内核模块目录名称。

```
  INSTALL drivers/usb/gadget/legacy/gadgetfs.ko
  INSTALL drivers/usb/gadget/libcomposite.ko
  INSTALL drivers/usb/serial/ftdi_sio.ko
  INSTALL drivers/usb/serial/option.ko
  INSTALL drivers/usb/serial/usb_wwan.ko
  INSTALL drivers/usb/serial/usbserial.ko
  INSTALL drivers/video/fbdev/mxc/mxc_dcic.ko
  INSTALL fs/binfmt_misc.ko
  INSTALL fs/configfs/configfs.ko
  INSTALL fs/fat/msdos.ko
  INSTALL fs/isofs/isofs.ko
  INSTALL fs/nls/nls_iso8859-15.ko
  INSTALL fs/udf/udf.ko
  INSTALL lib/crc-ccitt.ko
  INSTALL lib/crc-itu-t.ko
  INSTALL lib/crc7.ko
  INSTALL lib/libcrc32c.ko
  INSTALL sound/core/snd-hwdep.ko
  INSTALL sound/core/snd-rawmidi.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  DEPMOD  4.1.15-g11f73a0
root@ubuntu:/home/alientek/IMX6/linux-imx-4.1.15-2.1.0#
```

图 4.4. 5 安装模块

同时可以看到 rootfs 内核模块已经安装上去了。使用时请注意，内核版本需要与模块同一个版本号。否则在内核启动时找不到内核模块。一般地同一源码同一次编译出来的内核与内核模块是同一个版本号。可以在系统启动后使用 uname -r 查看内核的版本号，再进入/lib/modules 目录下查看模块版本号（也就是目录名称）。如果不相同，那么直接启动系统进入串口终端后使用指令 mv 4.1.15-xxxx $(uname -r)重命名内核模块目录，再重新启动即可。

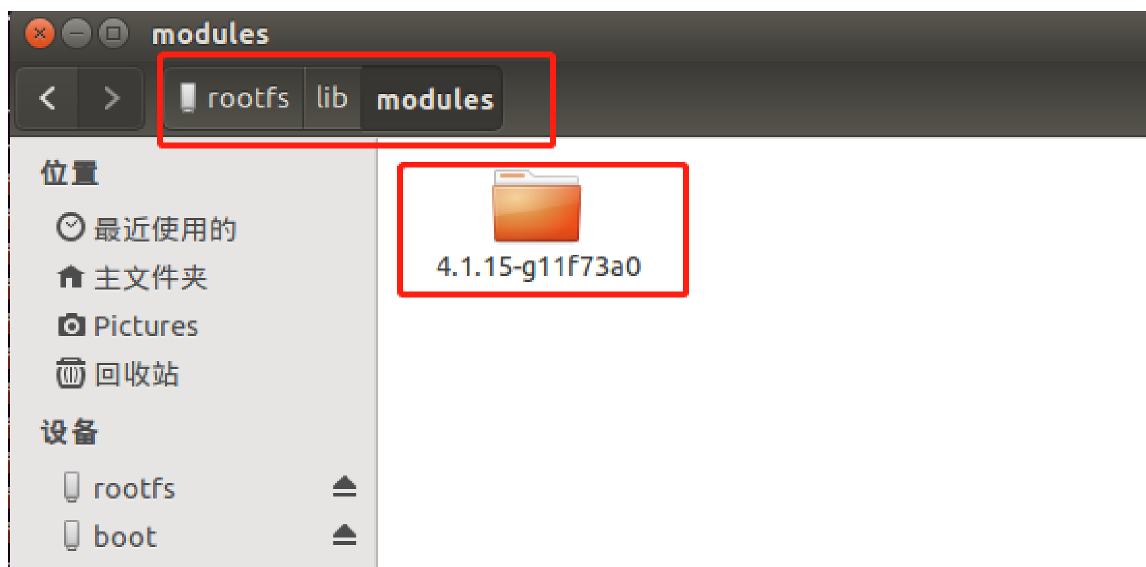


图 4.4. 6 安装模块到 rootfs 分区的/lib/modules 目录下

若你的系统是烧写在 emmc 设备或者 nandflash 设备则使用上面的指令先安装在任意的一个目录下，然后再用 U 盘拷贝（或者使用 scp 指令）这个模块目录到您的系统目录即可。SD 卡系统卡也是可以利用这种方法去更新内核模块。

4.5 编译 U-boot

将 1、例程源码->3、正点原子修改后的 Uboot 和 Linux->uboot-imx-2016.03-2.1.0-gxxxxxx-vx.x.tar.bz U-boot 源码解压到 Ubuntu 拷贝到虚拟机 Ubuntu 目录下，使用 tar 指令解压，格式是 tar jvxf + U-boot 源码压缩包。

进入 U-boot 源码目录如下

```
root@ubuntu:/home/alientek/IMX6/uboot-imx-2016.03-2.1.0# ls
api   board  common     configs  doc      dts       fs       Kbuild   lib       MAINTAINERS  Makefile  post    scripts          test
arch  cmd    config.mk  disk     drivers  examples  include  Kconfig  Licenses  MAKEALL      net       README  snapshot.commit  tools
root@ubuntu:/home/alientek/IMX6/uboot-imx-2016.03-2.1.0#
```

编辑编译脚本，使用 vi/vim 指令编辑一个名称为 build.sh 的脚本，拷贝以下内容到该脚本。按 i 进入插入模式，右键进行粘贴内容，粘贴成功后按 ESC 键退出编辑模式，然后输入:wq 保存并退出。

USER# vi build.sh

```
#!/bin/bash
make distclean
#配置 256MB DDR，  Nand flash 设备，从 SD 卡启动所用的 U-boot
make mx6ull_14x14_ddr256_nand_sd_defconfig
make all -j16
#将 u-boot.imx 重命名为 u-boot-imx6ull-ddr256-nand-sd.imx，下同
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-nand-sd.imx
#配置 512MB DDR ，Nand flash 设备，从 SD 卡启动所用的 U-boot
make mx6ull_14x14_ddr512_nand_sd_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-nand-sd.imx
#配置 256MB DDR， eMMC 设备，从 eMMC/SD 卡启动所用的 U-boot
make mx6ull_14x14_ddr256_emmc_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-emmc.imx
#配置 512MB DDR， eMMC 设备，从 eMMC/SD 卡启动所用的 U-boot
make mx6ull_14x14_ddr512_emmc_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-emmc.imx
#配置 256MB DDR ，Nand flash 设备，从 Nand flash 启动所用的 U-boot
make mx6ull_14x14_ddr256_nand_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-nand.imx
#配置 512MB DDR ，Nand flash 设备，从 Nand flash 启动所用的 U-boot
make mx6ull_14x14_ddr512_nand_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-nand.imx
```

```
alientek@ubuntu: ~/IMX6/uboot-imx-2016.03-2.1.0
#!/bin/bash
make distclean
make mx6ull_14x14_ddr256_nand_sd_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-nand-sd.imx
make mx6ull_14x14_ddr512_nand_sd_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-nand-sd.imx
make mx6ull_14x14_ddr256_emmc_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-emmc.imx
make mx6ull_14x14_ddr512_emmc_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-emmc.imx
make mx6ull_14x14_ddr256_nand_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr256-nand.imx
make mx6ull_14x14_ddr512_nand_defconfig
make all -j16
mv u-boot.imx u-boot-imx6ull-14x14-ddr512-nand.imx
~
~
~
~
:wq                       输入：wq保存退出
```

图 4.5. 1 编辑编译 U-boot 脚本

使用 chmod 指令修改 build.sh 的权限。

USER# chmod u+x build.sh

```
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ chmod u+x build.sh
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ ls
api   board     cmd     config.mk  disk  drivers  examples  include  Kconfig  Licenses     MAKEALL   net   README   snapshot.commit  tools
arch  build.sh  common  configs    doc   dts      fs        Kbuild   lib      MAINTAINERS  Makefile  post  scripts  test
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$
```

图 4.5. 2 赋予脚本可执行权限

使用 source 指令使能当前终端交叉编译环境变量（切换终端和用户需要重新执行下面这条指令）

USER# source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

```
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$
```

图 4.5. 3 使能环境变量

直接在当前路径下执行该脚本进行编译多个不同类型设备与及启动设备所用的 U-boot 镜像文件。

USER# ./build.sh

下面为编译 U-boot 的部分截图

```
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ ./build.sh
  CLEAN   tools
  CLEAN   tools/lib tools/common
  CLEAN   include/bmp_logo.h include/bmp_logo_data.h u-boot.cfg u-boot.lds
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated
  CLEAN   .config include/autoconf.mk include/autoconf.mk.dep include/config.h
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

图 4.5.4 编译 U-boot 的部分截图

编译成功如下图，会编译出 6 种不同配置与启动方式不同的 U-boot 镜像文件，根据个人的开发板配置类型选择不同的 U-boot 镜像烧写即可。

```
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ ls
api       disk      Kconfig      README           u-boot.cfg                          u-boot.map
arch      doc       lib          scripts          u-boot-imx6ull-14x14-ddr256-emmc.imx    u-boot-nodtb.bin
board     drivers   Licenses     snapshot.commit  u-boot-imx6ull-14x14-ddr256-nand.imx    u-boot.srec
build.sh  dts       MAINTAINERS  System.map       u-boot-imx6ull-14x14-ddr256-nand-sd.imx u-boot.sym
cmd       examples  MAKEALL      test             u-boot-imx6ull-14x14-ddr512-emmc.imx
common    fs        Makefile     tools            u-boot-imx6ull-14x14-ddr512-nand.imx
config.mk include   net          u-boot           u-boot-imx6ull-14x14-ddr512-nand-sd.imx
configs   Kbuild    post         u-boot.bin       u-boot.lds
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$
```

图 4.5.5 U-boot 源码顶层目录下的 u-boot 镜像文件

4.6 编译 Qt 工程

说明：编译 Qt 工程，这里只编译已经调试好的 Qt 工程，编译出可以在我们 IMX6 开发板上可以执行的文件。这里重复了我们正点原子的 IMX6 Qt Creator 交叉编译环境搭建里面的第 3 节交叉编译 Qt 工程。

拷贝 Qt 工程到 Ubuntu 虚拟机，例如本人拷贝 Qt web 浏览器工程（读者可以随意拷贝一个工程）光盘路径为 1、例程源码->9、Qt 综合例程源码->QWebBrowser 到本人的虚拟机里，使用 cd 命令进入到这个工程目录里，可以看到这一级目录有个 QWeBrowser.pro 工程项目文件。

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ ls
icon  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  QWebBrowser.pro  resource.qrc  webpage.cpp  webpage.h
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.1 查看工程文件“*.pro”

使用 source 指令使能当前终端交叉编译环境变量（切换终端和用户需要重新执行下面这条指令），如果您在上文已经使能过环境变量就不用再执行下面这一步了。

USER# source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.2 使能环境变量

执行指令 qmake 生成 Makefile 文件，可以直接在当前目录执行 qmake，或者写成 qmake + 工程名称。如 qamke QWebBrowser.pro

USER# qmake

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ qmake
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ ls
icon  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  Makefile  QWebBrowser.pro  resource.qrc  webpage.cpp  webpage.h
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.3 执行 qmake 生成 Makefile

直接执行 make，编译这个工程。

USER# make

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ make
/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/uic mainwindow.ui -o ui_mainwindow.h
arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -c -pipe  -O2 -DLINUX=1 -std=gnu++0x -Wall -W -D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WEBKITWIDGETS_LIB -DQT_WIDGETS_LIB -DQT_WEBKIT_LIB -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LIB -I. -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5 -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKitWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKit -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtGui -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtNetwork -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtCore -I. -I. -I/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++ -o main.o main.cpp
arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -c -pipe  -O2 -DLINUX=1 -std=gnu++0x -Wall -W -D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WEBKITWIDGETS_LIB -DQT_WIDGETS_LIB -DQT_WEBKIT_LIB -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LIB -I. -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5 -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKitWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKit -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtGui -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtNetwork -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtCore -I. -I. -I/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++ -o mainwindow.o mainwindow.cpp
arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -c -pipe  -O2 -DLINUX=1 -std=gnu++0x -Wall -W -D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WEBKITWIDGETS_LIB -DQT_WIDGETS_LIB -DQT_WEBKIT_LIB -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LIB -I. -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5 -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKitWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroot
```

图 4.6.4 执行 make 编译工程

编译成功如下图，会在当前目录下生成一个可执行文件，可以直接拷贝到我们 IMX6 开发板直接在程序目录下执行./QwebBrowser 在开发板上跑起来。

```
+0x -Wall -W -D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WEBKITWIDGETS_LIB -DQT_WIDGETS_LIB -DQT_WEBKIT_LIB -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LIB -I. -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5 -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKitWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWidgets -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtWebKit -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtGui -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtNetwork -isystem /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtCore -I. -I. -I/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++ -o moc_webpage.o moc_webpage.cpp
arm-poky-linux-gnueabi-g++  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed -Wl,-O1 -Wl,-rpath-link,/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib -o QWebBrowser main.o mainwindow.o webpage.o qrc_resource.o moc_mainwindow.o moc_webpage.o   -lQt5WebKitWidgets -lQt5Widgets -lQt5WebKit -lQt5Gui -lQt5Network -lQt5Core -lGLESv2 -lEGL -lpthread
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ ls
icon      main.o          mainwindow.h  mainwindow.ui  moc_mainwindow.cpp  moc_webpage.cpp  qrc_resource.cpp  QWebBrowser      resource.qrc     webpage.cpp  webpage.o
main.cpp  mainwindow.cpp  mainwindow.o  Makefile       moc_mainwindow.o    moc_webpage.o    qrc_resource.o    QWebBrowser.pro  ui_mainwindow.h  webpage.h
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.5 编译成功查看编译出来的文件

使用 file 指令查看生成的可执行文件的属性，可以看到 ARM 字样，说明可以在我们的 IMX6 开发板上跑的。

USER# file QWebBrowser

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ file QWebBrowser
QWebBrowser: ELF 32-bit LSB executable, ARM, EABI5 version 1 (GNU/Linux), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=62dad6690304311acb3452ad7356030d25f7543a, not stripped
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.6 查看文件信息

使用 make distclean 指令可以直接清除编译生成的*.o 文件、Makefile、生成的*.h 与生成的可执行文件。再使用 ls 查看当前工程目录，又变成一个干净原始的工程目录了。

USER# make distclean

```
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ make distclean
rm -f qrc_resource.cpp
rm -f moc_mainwindow.cpp moc_webpage.cpp
rm -f ui_mainwindow.h
rm -f main.o mainwindow.o webpage.o qrc_resource.o moc_mainwindow.o moc_webpage.o
rm -f *~ core *.core
rm -f QWebBrowser
rm -f Makefile
alientek@ubuntu:~/QWebBrowser/QWebBrowser$ ls
icon  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  QWebBrowser.pro  resource.qrc  webpage.cpp  webpage.h
alientek@ubuntu:~/QWebBrowser/QWebBrowser$
```

图 4.6.7 清理工程

4.7 编译单独的 c 文件

本文新建一个 test 文件夹，在 test 文件夹下编辑一个 main.c 文件，使用 vi/vim 指令，复制下面的内容。程序功能是打印一句“hello world!”，拷贝以下内容到 vi/vim 编辑器窗口里。按 i 进入插入模式，右键进行粘贴内容，粘贴成功后按 ESC 键退出编辑模式，然后输入:wq 保存并退出。

USER# vi main.c

```
#include <stdio.h>
int main(void)
{
		printf("hello world!\n");
		return 0;
}
```

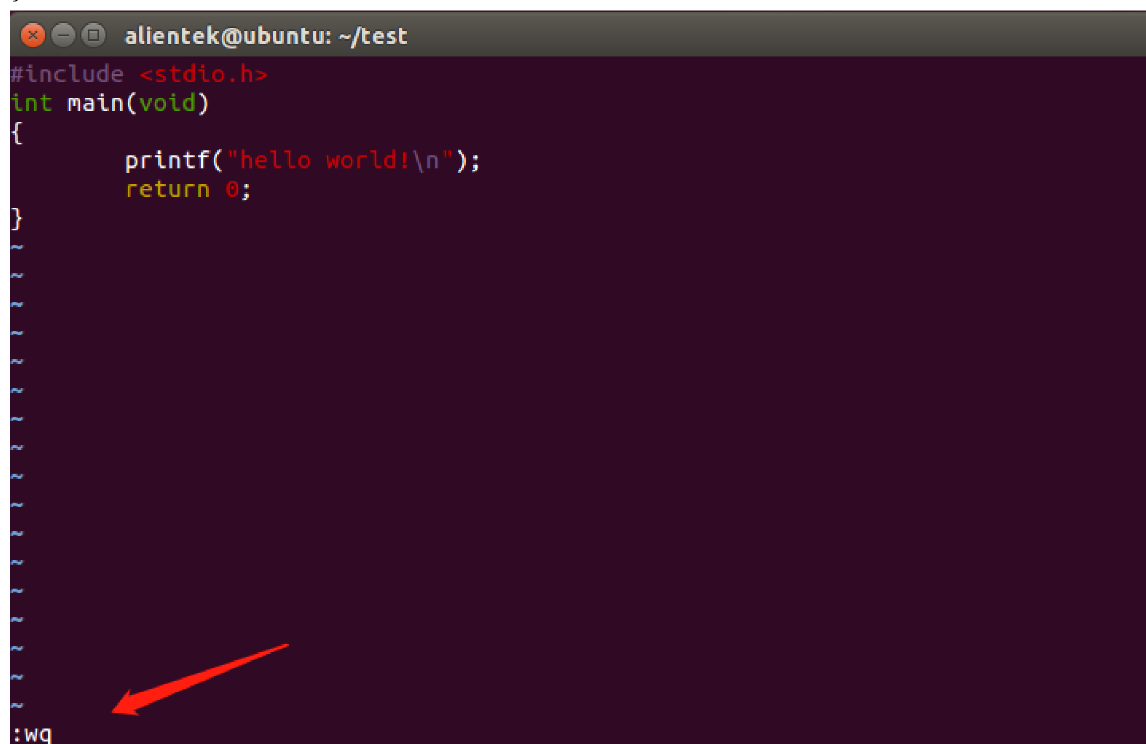


图 4.6.8 编辑 c 文件

编辑完成保存后，用 ls 指令查看如下

```
alientek@ubuntu:~/test$ vi main.c
alientek@ubuntu:~/test$ ls
main.c
alientek@ubuntu:~/test$
```

图 4.6.9 编辑的文件

使用 source 指令使能当前终端交叉编译环境变量（切换终端和用户需要重新执行下面这条指令），如果您在上文已经使能过环境变量就不用再执行下面这一步了。

USER# source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

```
alientek@ubuntu:~/test$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~/test$
```

图 4.6. 10 使能环境变量

使能环境变量后使用 env 指令查看当前环境变量，可以看到编译工具链的 sdk 路径以及参数已经赋值给“CC”，所以我们可以直接调用来编译 main.c 文件。

USER# env

```
LOGNAME=alientek
OE_QMAKE_MOC=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/moc
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3juRGIrtNI
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/
QT4_IM_MODULE=fcitx
OE_QMAKE_QDBUSCPP2XML=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qdbuscpp2xml
OECORE_ACLOCAL_OPTS=-I /opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/x86_64-pokysdk-linux/usr/share/aclocal
PKG_CONFIG_PATH=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/pkgconfig
LESSOPEN=| /usr/bin/lesspipe %s
ARCH=arm
RANLIB=arm-poky-linux-gnueabi-ranlib
OE_QMAKE_CFLAGS=
INSTANCE=Unity
TEXTDOMAIN=im-config
UPSTART_JOB=unity-settings-daemon
CROSS_COMPILE=arm-poky-linux-gnueabi-
CC=arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=fcitx
OE_QMAKE_LIBDIR_QT=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib
OBJDUMP=arm-poky-linux-gnueabi-objdump
LESSCLOSE=/usr/bin/lesspipe %s %s
LC_TIME=en_US.UTF-8
OE_QMAKE_LDFLAGS=-Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed
SDKTARGETSYSROOT=/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi
TEXTDOMAINDIR=/usr/share/locale/
LC_NAME=en_US.UTF-8
XAUTHORITY=/home/alientek/.Xauthority
COLORTERM=gnome-terminal
_=/usr/bin/env
alientek@ubuntu:~/test$
```

图 4.6.11 查看环境变量 CC 的值

编译 main.c 文件，“-o”的意思是指定编译后输出的文件名称，此处输出为“main”。

USER# $CC main.c -o main

或者写成下面这样来编译 main.c 文件也是可以的

USER# arm-poky-linux-gnueabi-gcc -march=armv7ve -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=$SDKTARGETSYSROOT main.c -o main

```
alientek@ubuntu:~/test$ ls
main.c
alientek@ubuntu:~/test$ $CC main.c -o main
alientek@ubuntu:~/test$ ls
main  main.c
alientek@ubuntu:~/test$ rm main
alientek@ubuntu:~/test$ arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=$SDKTARGETSYSROOT main.c -o main
alientek@ubuntu:~/test$ ls
main  main.c
alientek@ubuntu:~/test$
```

图 4.6.12 查看编译出来的文件

使用 file 指令查看生成的可执行文件的属性，可以看到 ARM 字样，说明可以在我们的 IMX6 开发板上跑的。

USER# file main

```
alientek@ubuntu:~/test$ file main
main: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked
, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=9442
5819984d8a1b073f1a50b90d83c118b421ab, not stripped
alientek@ubuntu:~/test$
```

图 4.6.13 查看编译出来的文件的信息

本文使用 scp 指令拷贝 main 文件到我们 IMX6 开发板的/home/root 目录下（备注 scp 指令用法如果读者不会，可以仿照下面来写，主要我们的主机 ip 地址可能不同）。如果不会用 scp 指令，那么将用 U 盘来复制这个可执行文件到开发板，用 chmod u+x main 修改文件权限就可以执行了。

```
alientek@ubuntu:~/test$ scp main root@192.168.1.115:/home/root
The authenticity of host '192.168.1.115 (192.168.1.115)' can't be established.
RSA key fingerprint is 9a:f9:30:81:37:f6:99:fa:5b:38:cf:18:97:7c:81:94.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.115' (RSA) to the list of known hosts.
main                                          100% 9468     9.3KB/s   00:00
alientek@ubuntu:~/test$
```

图 4.6.14 拷贝编译出来的可执行文件到开发板文件系统

在/home/root 目录下执行该文件。

USER# ./main

```
root@ALIENTEK-IMX6:~# ./main
hello world!
root@ALIENTEK-IMX6:~#
```

图 4.6.15 直接在当前目录下执行该文件，打印出“hello world”信息

第五章 ALIENTEK I.MX6U Qt 应用程序

5.1 Qt 源码路径

在开发板光盘目录下 1、例程源码->9、Qt 综合例程源码下。例程仅供参考，用户可参考来自行修改与编译。正点原子会继续更新 Qt 程序和优化 Qt 程序。

5.2 Qt 程序路径

在文件系统/opt/qt5.5.1/apps 下。程序仅供参考。后期正点原子会优化 Qt 应用程序，敬请期待！

5.3 Qt 环境变量设置

在/etc/profile 文件里，后续详细介绍。

5.4 Qt 程序自启动

在/etc/rc.local 这个文件里，比较简单。

附录 A